Médiation de données sémantique dans SenPeer, un système pair-à-pair de gestion de données

David Célestin Faye

École Centrale de Nantes Université de Nantes École des Mines de Nantes

ÉCOLE DOCTORALE STIM
« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX »

Année 2007

Médiation de données sémantique dans SenPeer, un système pair-à-pair de gestion de données

---

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE NANTES

Discipline : INFORMATIQUE

présentée et soutenue publiquement par

David Célestin FAYE

le 22 Octobre 2007

à la Faculté des sciences de l'Université de Nantes

devant le jury ci-dessous

Rapporteurs	:	Zohra BELLAHSENE, Professeur	Université de Montpellier II
		Kokou YETONGNON, Professeur	Université de Bourgogne
Examinateurs	:	Mary Teuw NIANE , Professeur	Université Gaston Berger de Saint-Louis
		Patrick VALDURIEZ , Directeur de Recherches	INRIA
		Gilles NACHOUKI , Maître de Conférences	Université de Nantes

Directeurs de thèse : Patrick VALDURIEZ, Mary Teuw NIANE
Encadrant de thèse : Gilles NACHOUKI
Laboratoire : LINA (Laboratoire d'Informatique de Nantes Atlantique)

N° ED 0366-XXX

MÉDIATION DE DONNÉES SÉMANTIQUE DANS SENPEER, UN SYSTÈME PAIR-À-PAIR DE GESTION DE DONNÉES

Semantic data mediation in SenPeer, a peer-to-peer data management system

David Célestin FAYE

favet neptunus eunti

Université de Nantes

David Célestin FAYE

Médiation de données sémantique dans SenPeer, un système pair-à-pair de gestion de données

x+144 p.

Ce document a été préparé avec LaTeX2$_\varepsilon$ et la classe `these-IRIN` version 0.92 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-IRIN` est disponible à l'adresse :

`http://login.irin.sciences.univ-nantes.fr/`

Impression : RapportThese.tex – 14/10/2007 – 18:25

Révision pour la classe : $Id: these-IRIN.cls,v 1.3 2000/11/19 18:30:42 fred Exp

Remerciements

Je tiens à remercier très sincèrement Patrick Valduriez d'avoir bien voulu me proposer ce sujet de thèse dans un contexte particulièrement difficile marqué par un changement de sujet de thèse et de directeur. Merci infiniment pour les orientations, et la rigueur scientifique. Merci aussi d'avoir accepté le principe de la thèse en alternance entre le Sénégal et la France, malgré mes tâches d'enseignant à l'Université Gaston Berger de Saint-Louis.

Mes remerciements vont à l'endroit de Mary Teuw Niane, Recteur de l'Université Gaston Berger de Saint-Louis, qui a bien voulu être le co-directeur de cette thèse pour la partie sénégalaise de la cotutelle. Dans le même élan, je remercie les dirigeants de l'institution qui ont bien voulu me libérer par moments pour que je puisse séjourner au LINA pour les besoins de ma thèse.

Je tiens à exprimer mes plus vifs remerciements à mon encadrant Gilles Nachouki qui m'a aidé à faire mes premiers pas dans la recherche, qui me suit depuis mon DEA à Nantes et qui a vraiment tout mis en oeuvre pour que je puisse faire cette thèse. Merci infiniment pour la disponibilité, la patience, les remarques et suggestions pertinentes durant toute la thèse. Merci d'avoir été à mes côtés dans les moments les plus difficiles.

Je remercie les membres de mon jury de thèse : les rapporteurs Zohra Bellahsene, de l'Université de Montpellier II et Kokou Yetongnon, de l'Université de Bourgogne.

Je remercie le Service de Coopération et d'Action culturelle de l'Ambassade de France au Sénégal pour son soutien financier qui m'a permis d'effectuer mes séjours de thèse en alternance entre le LINA de Nantes et le LANI de l'Université Gaston Berger de Saint-Louis.

Plus personnellement je remercie tous mes amis de Nantes qui m'ont beaucoup soutenu : Guillaume, Freddy, Antoine, Nassima, Cédric, Stéphanie, Vince, Soizic, Romain, Mathilde, et tous ceux que j'ai oubliés de citer. Merci à Maman Françoise et à Papa Michel, mes parents sur Nantes.

Enfin, je tiens remercier Tatiana, ma femme, pour son soutien infini.

A toutes et à tous, merci.

Sommaire

1 Introduction **1**
2 État de l'art **7**
3 Médiation de données sémantique **37**
4 Traitement des requêtes **69**
5 Validation **101**
6 Conclusion **115**

Bibliographie **119**
Liste des tableaux **129**
Table des figures **131**
Table des matières **133**

CHAPITRE 1

Introduction

La société de l'information demande un accès efficace à l'information disponible, information qui est souvent hétérogène et distribuée. Ces dernières années, il a été noté une croissance drastique des informations électroniques échangées. Chaque utilisateur se connecte à Internet et a accès à une quantité importante d'informations en ligne. Cette information est la plupart du temps disséminée à travers un grand nombre de sources de données. L'utilisateur, en plus de n'être intéressé que par une petite portion de cette information, souhaite consacrer le minimum de temps et de ressources pour son acquisition. Dans le but de partager efficacement cette information, plusieurs solutions techniques ont été proposées. Le Web et les réseaux Pair-à-Pair (P2P) ont permis de mettre en place des moyens simples de partage de données entre les utilisateurs tout en se limitant cependant à la recherche par mots-clés.

Dans le but d'organiser des données dispersées sur différents sites, le concept de base de données distribuée a été introduit. Une base de données distribuée est définie comme une collection multiple de bases de données logiquement liées et distribuées sur un réseau d'ordinateurs. Un système de base de données distribuée est le logiciel qui permet la gestion de la base de données distribuée. Son rôle principal est de rendre la distribution transparente à l'utilisateur en masquant les détails fastidieux liés à la gestion des données. Un tel système, dans sa forme la plus simple, est constitué d'un serveur central supportant un schéma global et offrant les services d'une base de données distribuée, traitement de requêtes distribuées et gestion de la consistence des données entre autres. Le principe fondamental derrière la gestion des données est l'indépendance des données. Cette indépendance permet aux applications et aux utilisateurs de partager des données au niveau conceptuel tout en ignorant les détails liés à l'implémentation. Cette approche fournit ainsi un support pour la gestion des schémas, l'expression de requêtes de haut niveau, le traitement et l'optimisation automatique de requêtes.

Dans cette direction, l'infrastructure Pair-à-Pair (P2P) est un paradigme émergent et offrant de nouvelles opportunités pour la mise en place de systèmes distribués à grande échelle en innovant l'approche des bases de données distribuées. Les systèmes P2P attirent de plus en plus l'attention grâce à leur ascension fulgurante occasionnée par la popularité des systèmes de partage de fichiers tels que Napster[81], Kazaa[57], Gnutella[91], pour ne citer que ceux-là. Ces systèmes ont la particularité de pouvoir interconnecter des millions d'utilisateurs dans le but de leur faire partager une quantité importante de données. Par conséquent, les systèmes P2P nécessitent la disponibilité d'un grand nombre d'ordinateurs autonomes (les pairs) qui mettent en commun leurs ressources et coopèrent pour le partage de données et de services. Contrairement au modèle Client/Serveur classique, un système P2P opère sans coordination centrale, et fournit par conséquent un environnement dynamique que les pairs peuvent joindre ou quitter à tout moment. Entre autres avantages, nous pouvons citer les communications directes et rapides entre pairs, l'auto-organisation, la décentralisation dans le stockage et le traitement des données ainsi que la possibilité de supporter un très grand nombre de pairs, tout en assurant la tolérance aux fautes.

Dès lors, pour venir à bout des limites des bases des données distribuées dans lesquelles le schéma central médiateur qui, en plus d'être un frein à l'évolution des schémas, complique aussi le partage de données, le couplage entre celles-ci et les systèmes P2P devient naturel. C'est ainsi qu'est apparu le

concept de système Pair-à-Pair de gestion de données communément appelé PDMS (*Peer Data Management System*) [41].

Dans le but de contribuer à la problématique de la gestion de données dans un contexte P2P, nous proposons le PDMS SenPeer. Son but principal est de fournir les outils nécessaires pour la médiation de données sémantique, le routage sémantique de requêtes et le traitement des ces requêtes dans un environnement P2P. Plus précisément, SenPeer permet le partage décentralisé de données entre un ensemble de pairs autonomes, chaque pair publiant une source de données pouvant se conformer à un modèle de données quelconque.

1.1 Motivations

Les systèmes Pair-à-Pair classiques tels que Napster[81] ou Kaaza[57] se sont illustrés par une description des nœuds par clés et une localisation des données par un routage de ces mêmes clés dans le réseau. En dépit de leur essor fulgurant, la plupart de ces systèmes présentent des limites notoires qui sont : le partage de fichiers uniquement, la recherche simple basée sur les mots-clés ou encore l'accès en lecture seule. Dans la plupart des cas, les techniques de recherche sont des conditions de sélection simple sur des couples attribut-valeur ou basées sur la comparaison de chaînes de caractères propre au domaine de la Recherche d'Informations (IR). D'autre part, la localisation de ces données est basée sur l'inondation du réseau par les requêtes, ce qui occasionne une consommation importante des ressources du réseau et des capacités de calcul disponibles. Ces limitations ne sont pas envisageables dans un environnement dynamique et distribué mettant en jeu un nombre important de pairs autonomes qui ont besoin de mécanismes d'échange et d'interrogation de données sémantiquement et syntaxiquement hétérogènes.

Récemment, les systèmes Pair-à-Pair de gestion de données communément appelés PDMS (*Peer Data Management System*) [41] ont vu le jour. Ils combinent la technologie Pair-à-Pair et celle des bases de données distribuées et s'appuient sur une description sémantique des sources de données pour aider à la formulation et le routage des requêtes à travers le réseau, mais aussi à l'intégration des résultats. Dans ce contexte, la problématique de la gestion des données dans un environnement P2P soulève de nouveaux défis dus à la taille du réseau, à l'autonomie des pairs mais aussi à leur volatilité.

La mise en place d'un PDMS nécessite la prise en compte des exigences suivantes :

Autonomie : chaque pair devra être en mesure de joindre ou de quitter le réseau à tout moment. Il devra aussi être en mesure de contrôler et de gérer ses données indépendamment du reste du système.

Médiation de données : La maintenance d'un schéma global est un frein à l'extensibilité du réseau, vu son aspect dynamique. Les sources étant hétérogènes, il faut trouver un moyen pour réconcilier les différentes représentations de façon dynamique.

Réécriture des requêtes : Les langages d'interrogation des pairs n'étant pas forcément les mêmes, il faut trouver un mécanisme permettant, étant donné une requête formulée avec un langage donné sur le schéma d'un pair, de reformuler cette dernière sur le schéma d'autres pairs avec les langages d'interrogation de ces derniers. Ces langages d'interrogation devraient aussi permettre à l'utilisateur de décrire les données recherchées dans un niveau de détail approprié.

Efficacité : L'utilisation rationnelle et efficace des ressources distribuées (capacité de calcul et de stockage, bande passante) devrait amoindrir les coûts et résulter en une capacité à traiter un plus grand nombre de requêtes en un instant donné.

Qualité de service : Ce paramètre fait référence à la façon dont l'utilisateur perçoit l'efficacité du système, c'est-à-dire, l'aspect complet des résultats, la validité des données, la disponibilité des

données, le temps de réponse, etc.

Tolérance aux fautes : L'efficacité et la qualité de service devraient être préservées en dépit de failles émanant de la nature dynamique du système.

La problématique du partage de données dans un environnement P2P a attiré l'attention d'un nombre récent de projets[103][84][23][82][45]. Ces PDMS utilisent essentiellement la connaissance intentionnelle véhiculée par les schémas pour réconcilier les données, mais aussi pour aider à la localisation des pairs pertinents pour une requête donnée. Des langages de requêtes riches sont aussi supportés. Nous pouvons identifier deux tendances dans les PDMS :

- D'un côté le schéma est supposé être homogène et les principaux défis sont le traitement de requêtes et la génération des plans supervisant l'exécution des requêtes.
- De l'autre côté, les schémas sont supposés être hétérogènes et par conséquent l'intégration des schémas devient le principal défi. Dans ce dernier cas, des correspondances sémantiques sont utilisées pour passer d'un schéma à l'autre. Cependant, la capacité à passer à l'échelle devient restreinte.

Quelques uns des systèmes se focalisant sur les schémas homogènes passent mieux à l'échelle. Par ailleurs, il est très difficile de comparer l'efficacité de ces systèmes, d'abord parce qu'il n'existe pas de repère standard et ensuite parce que les modèles de données et l'expressivité des langages de requêtes varient largement d'un système à l'autre. Nous avons néanmoins noté que la plupart des PDMS ne permettent pas le partage de données (de n'importe quel domaine) décrites par des modèles de données différents[103][84][23][82][45]. Dans certains cas, leur mise en place est lourde car elle nécessite un certain nombre de correspondances sémantiques[28][14], ou parfois même, fait appel à l'intervention humaine[7][84], ce qui n'est pas envisageable dans un environnement distribué mettant en jeu un nombre important de pairs.

Pour mieux illustrer le problème auquel nous nous intéressons, considérons le scénario motivant suivant lié à la mise en valeur de la vallée du fleuve sénégal. Le fleuve sénégal délimite une frontière naturelle entre le Mali, la Mauritanie et le Sénégal. Toute la région du fleuve constitue une zone agricole, d'élevage et de pêche partagée par ces trois pays frontaliers. La mise en valeur de la vallée du fleuve fait intervenir, depuis des années, des experts de divers organismes (OMVS - Organisation pour la Mise en valeur de la Vallée du Fleuve Sénégal, ISRA - Institut Sénégalais de Recherche Agronomique, SAED - Société d'Aménagement et d'Exploitation des terres du Delta, OMS - Organisation Mondiale de la Santé, etc.) de différents domaines de compétences (hydraulique, activités agricoles, recherche agronomique, santé, etc.) mais aussi localisés dans ces différents pays. Tous ces experts mènent des travaux qui aboutissent généralement à la production et à l'exploitation de gros volumes de données. La gestion et l'exploitation des données sont loin d'être satisfaisantes à cause de leur distribution, hétérogénéité, volume et appartenance[10]. En effet, le système actuel de partage de données est basé sur un serveur central localisé à Dakar. Ce serveur coordonne toutes les sources de données de la sous-région et sa défaillance entraînerait l'effondrement de tout le réseau de partage de données. De plus, il ne permet que l'échange de bases de données relationnelles. La politique de mise en valeur passe par une exploitation rationnelle des données existantes. Il est donc capital de fournir aux producteurs et consommateurs de données de puissants moyens d'intégration, de gestion, de diffusion et de recherche des données existantes. Cela va ainsi permettre aux décideurs de tirer le maximum d'informations pertinentes à partir de ces données. Nous pensons qu'une plate-forme logicielle de type PDMS semble bien adaptée à ce scénario. Dans un réseau de partage de données comme celui de la figure 1.1, chaque expert ou pair devra, en toute autonomie, être en mesure de :

- fournir des données de nature diverse en vue de les partager avec les autres experts quelque soit

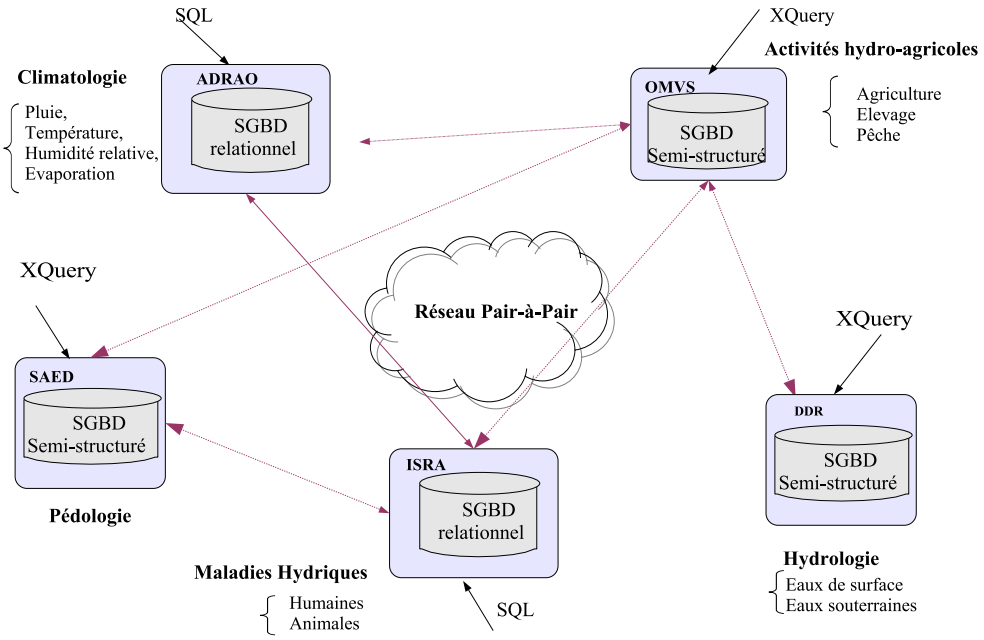


Figure 1.1 – Partage de données du fleuve en présence de plusieurs modèles de données.

son pays ou son domaine de compétence et de les lier à celles existantes ;

- effectuer la recherche de données pertinentes sur celles proposées par les autres experts. Il peut choisir d'étendre ses données avec les nouvelles données trouvées.

Les données relatives à un domaine étant réparties entre différents acteurs, il y a un besoin de répondre à des requêtes complexes du type :

- (R1) *Quel est le taux de bilharziose quand le débit des eaux est de 640* m^3/s*?*
- (R2) *Quel est le rendement des cultures sous pluie quand la température moyenne saisonnière est de 30*$^\circ$*?*
- (R3) *Quelles sont les caractéristiques des sols sur lesquels le riz est cultivé?*

Ces requêtes nécessitent une médiation sémantique tenant compte des liens complexes pouvant exister entre les différentes données des experts. Par exemple, le traitement de la requête (R1) va impliquer les spécialistes des maladies hydriques humaines et ceux de l'hydraulique, mais aussi tenir en compte le fait que les données d'un même domaine aient été mises en place par différents acteurs. Le problème est d'autant plus difficile que ces derniers sont autorisés à fournir des données basées sur des modèles de données différents (relationnel, XML, etc.). Dans ce cas, une requête sur le PDMS sera posée sur le schéma d'un pair spécifique avec son propre langage d'interrogation SQL, XQuery, etc. Plus généralement, les questions suivantes devront être répondues :

1. Comment les pairs représentent-ils leurs sources de données?
2. Comment les pairs informent-ils le reste du réseau de leurs données?
3. Comment résoudre l'hétérogénéité syntaxique et sémantique des pairs?
4. Comment les pairs routent-ils leurs requêtes?
5. Comment les pairs traitent-ils les requêtes?
6. Comment les plans de requêtes distribuées sont-ils optimisés?

1.2 Contributions

Considérant les questions mentionnées ci-dessus, nous présentons les aspects liés à la conception du système SenPeer[34] pour le partage de données dans un environnement P2P. Nous n'avons pas l'intention, dans ce travail, de produire un substitut au travail établi dans les domaines de l'intégration de données et du traitement de requêtes distribuées. En effet, nous avons besoin d'utiliser certains de ces travaux à chaque fois qu'ils s'appliquent aux problèmes qui se posent à nous pendant le partage des ressources (médiation de données, optimisation de requêtes, etc.). Cependant, nous avons constaté que la plupart des PDMS n'insistent pas sur le cas où le réseau de partage de données autorise les pairs à publier des données conformes à un modèle de données quelconque et par conséquent l'utilisation de plusieurs langages de requêtes. De plus, il est supposé, dans certain cas, une homogénéité de schémas étant donné que les données sont parfois décrites par rapport à des ontologies de référence, ce qui empêche la réutilisation. Par conséquent, notre approche devient complémentaire à celles déjà proposées. Le but de notre travail est donc la médiation de données et le traitement des requêtes à l'échelle d'un système P2P de gestion de données distribuées supportant plusieurs modèles de données et autorisant plusieurs schémas/ontologies.

Nous faisons les contributions suivantes :

- Nous passons en revue l'état de l'art des approches récentes de bases de données distribuées et des systèmes pair-à-pair de gestion de données distribuées. Les dimensions à prendre en compte dans la conception d'un PDMS sont aussi introduites. Il existe une panoplie de systèmes et d'approches, chacun avec ses compromis et ses spécificités et, par conséquent, ils sont difficilement comparables de façon stricte. Ces approches sont classifiées suivant les dimensions à prendre en compte dans la mise en place d'un PDMS, mais aussi suivant leurs capacités en termes de médiation de données et de traitement de requêtes.
- Nous proposons un processus d'intégration de sources de données hétérogènes détenues les pairs du réseau en l'absence de schéma global et en présence de différents modèles de données et langages de requêtes. Dans le but de permettre l'interopérabilité sémantique, notre approche se base sur un modèle pivot interne capturant la sémantique et la structure des schémas dans leur modèles d'origines et ayant pour but de faciliter la découverte des correspondances sémantiques entre les schémas des pairs désireux de fournir ou d'acquérir des données. Ces correspondances sont établies grâce à un ensemble de mesures de similarité sémantique tenant en compte aussi bien la structure des éléments des schémas que la connaissance linguistique qu'ils véhiculent.
- Nous introduisons la notion d'expertise de pair en tant que résumé succincte de la connaissance des pairs. Les expertises des différents pairs, en combinaison avec les correspondances sémantiques, sont à la base d'une topologie sémantique au dessus du réseau physique actuel, et dans laquelle les pairs sont organisés en communautés sémantiques autour de super-pairs et ceci en accord avec leurs thèmes d'intérêts. Cette topologie sémantique sert de support à un mécanisme de routage sémantique permettant une distribution intra et inter-communautés des requêtes uniquement au sous-ensemble de pairs pouvant y répondre, ce qui diminue de façon significative les efforts de traitement des requêtes.
- Nous proposons un processus de traitement de requêtes dans lequel les requêtes sont échangées grâce à un format commun d'échange de requêtes permettant d'éviter les multiples reformulations entre langages de requêtes. Un algorithme de réécriture sémantique de requête permet de reformuler correctement une requête formulée sur le schéma d'un pair avec son propre vocabulaire et son propre langage de requête, en une requête distante exécutable sur le schéma d'un pair lié, cette fois-ci avec le vocabulaire et le langage d'interrogation de ce dernier. Nous raffinons aussi

les concepts de réponse certaine et de contenance de requêtes dans le contexte pair-à-pair et en présence de communautés sémantiques.
- Nous décrivons le processus d'optimisation de requêtes distribuées impliquant une coopération entre les (super-)pairs participant à la requête. Les plans de requêtes sont générés à partir des informations renvoyées par l'algorithme de routage sémantique et leur optimisation dépend d'un modèle de coût prenant en compte un certain nombre de paramètres liés à la nature du réseau P2P. Cette optimisation est initiée au sein d'une communauté et elle est poursuivie dans les communautés impliquées dans le traitement de la requête, rendant ainsi les phases de génération de plan et de routage imbriquées.
- Pour évaluer nos algorithmes, nous avons mis en place un simulateur pour notre PDMS. Nous avons conduit des expériences en rapport avec les techniques de découverte de correspondances, mais aussi sur l'impact de l'organisation des pairs en communauté sémantiques sur la distribution des requêtes intra et inter-communautés. Nous avons aussi évalué le processus d'optimisation de requêtes proposé.

1.3 Plan de la thèse

La thèse est composée de 6 chapitres. Le reste du manuscrit est organisé comme suit.

Le chapitre 2 présente l'état de l'art des travaux concernant les systèmes pair-à-pair de gestion de données distribuées. Nous rappelons le problème du partage de données ainsi que les différentes technologies de partage de données telles que les bases de données distribuées et les systèmes P2P de gestion de données distribuées en insistant sur leurs différents aspects. Nous passons aussi en revue quelques systèmes proposés dans la littérature.

Dans le chapitre 3, nous présentons l'architecture du système SenPeer ainsi que le processus de réconciliation sémantique. La topologie physique du réseau est introduite ainsi que la structure logique des nœuds du réseau. Nous posons aussi le problème de la médiation sémantique dans notre contexte ainsi que le processus de découverte des correspondances sémantiques établissant une topologie sémantique au dessus de la topologie physique.

Le chapitre 4 présente le processus de traitement de requêtes. Nous indiquons comment les requêtes sont formulées et routées sémantiquement en tenant compte de la topologie sémantique induite par le processus de réconciliation sémantique. L'algorithme de routage sémantique est illustré de même que la phase de mise sous forme algébrique des requêtes en accord avec les informations sur la localisation des nœuds pertinents pour la requête. L'optimisation des plans de requêtes par le biais d'heuristiques est introduite, ainsi que leur distribution et exécution par le déploiement de canaux de communication appropriés.

Le chapitre 5 présente la validation par simulation des techniques proposées dans les chapitres précédents. Nous introduisons successivement les évaluations du processus de découverte de correspondances, du routage et de l'optimisation des requêtes. Pour chaque technique nous présentons le modèle de réseau utilisé pour la simulation ainsi que les métriques de performance utilisées avant de commenter les résultats obtenus.

Finalement, le Chapitre 6 résume nos travaux, les différentes contributions et les limites de notre approche. Nous donnons aussi quelques perspectives pour des travaux futurs.

CHAPITRE 2

État de l'art

2.1 Introduction

Dans ce chapitre, nous présentons les travaux de recherche en relation avec SenPeer. Nous passons en revue la problématique du partage de données ainsi que les différentes technologies proposées dans le passé. Dans un premier temps, nous présentons les bases de données distribuées dans certains de leurs aspects tels que le traitement des requêtes et la médiation sémantique de données. Nous citons aussi quelques projets de recherche tout en insistant sur les limites générales des bases de données distribuées. De nos jours, les systèmes pair-à-pair sont devenus un domaine de recherche à part entière et nous passons en revue leurs caractéristiques essentielles notamment les techniques d'indexation. Les systèmes P2P de gestion de données distribuées sont aussi introduits comme une convergence naturelle des systèmes P2P et des bases de données distribuées. Ces systèmes deviennent de plus en plus attractifs vu le nombre croissant de projets de recherche s'y intéressant. Nous rappelons les dimensions à prendre en compte dans la conception d'un système Pair-à-Pair de gestion de données distribuées ainsi que les techniques de découverte correspondances sémantiques et de traitement de requêtes à l'échelle de ces systèmes. Nous terminons par une présentation et une classification de quelques PDMS proposés dans la littérature en accord avec les dimensions à prendre en compte dans la conception d'un PDMS.

2.2 Le problème du partage de données

Le problème du partage de données a été posé depuis un certain nombre d'années. De nos jours, aussi bien les entreprises que les communautés scientifiques et les gouvernements éprouvent un large besoin de rendre accessible leurs données, mais aussi d'accéder à des données d'autres structures. Le défi majeur dans ce cas est de venir à bout de l'hétérogénéité des données, connue aussi sous le nom *d'hétérogénéité de schémas*. En effet, les sources d'informations sont nombreuses et diversifiées (bases de données relationnelles et objets, Pages Web, fichiers, etc.) ainsi que les modes de consultation (façon de *formuler* la requête, manière de *présenter* les résultats).

Le problème de l'intégration de données consiste à fournir un accès (requêtes, parfois mise à jour) uniforme à une multitude de sources de données hétérogènes à travers une vue unifiée appelée schéma médiateur. L'uniformité implique que les sources soient transparentes aux utilisateurs. Ces derniers s'intéressent à comment spécifier ce qu'ils veulent plutôt que de penser à comment obtenir les résultats à leurs requêtes. Les sources sont hétérogènes en ce sens qu'elles peuvent s'appuyer sur différents modèles de données et schémas.

Un système de partage de données doit aussi avoir les propriétés suivantes [99]:

- ***Passage à l'échelle*** : le nombre de sources de données supportées doit être potentiellement élevé (centaines de milliers).

- ***Robustesse*** : l'indisponibilité de certaines sources ne doit affecter en rien ou doit avoir un effet minimal sur le fonctionnement global du système. En d'autres termes, les autres nœuds du système doivent pouvoir continuer à partager leurs données dans de pareilles situations.
- ***Autonomie*** : pour chaque source de données la jonction ou le départ du système doit être assez aisée et ne doit pas nécessiter une grande mobilisation des nœuds sources déjà connectés. Plus encore, les sources ont la latitude de changer la représentation de leurs données à tout moment.
- ***Langage de requêtes expressif*** : le langage de requêtes utilisé doit être suffisamment expressif pour décrire les données recherchées. Il doit en outre être similaire à ceux supportés par les bases de données.

2.3 Partage de documents

Le Web a connu une ascension fulgurante et est devenu très populaire ces dernières années. Ceci est essentiellement dû à la manière assez facile pour les utilisateurs de publier leurs informations sur le Web. Ces informations sont, pour la plupart, des documents textes. La dissémination de ces documents à travers ce vaste réseau a conduit à la mise en place d'un certain nombre de moteurs de recherche dont le rôle principal est de renvoyer à l'utilisateur un certain nombre de pages vérifiant des critères que ce dernier aura auparavant spécifié, plutôt que de naviguer dans la totalité du réseau. La technique utilisée est celle des systèmes de recherche d'informations et donc dirigée par des mots-clés. Le problème dans ce cas est le manque de sémantique, puisque les pages Web sont des fichiers plats et donc non-structurés, ce qui ne permet pas de supporter des langages de requêtes riches.

D'autre part, la technologie P2P a servi de support au développement de plusieurs applications de partage de fichiers très populaires tels que Napster[81], Gnutella[91] et Kazaa[57], pour ne citer que ceux-là. Dans ce type de système, chaque pair (ordinateur connecté au réseau P2P) peut joindre ou quitter le réseau quand il le souhaite, publier ou rechercher des fichiers. Comme dans le Web, nous constatons que les systèmes actuels supportent un nombre de fonctions limitées au partage de fichiers avec des techniques de recherche assez simples et se faisant par mots-clés. En effet, ces systèmes traitent les fichiers comme des entités et ne gèrent pas leurs contenus.

Dans ces deux cas, la sémantique des données est totalement ignorée. Cette ignorance de la sémantique limite les ambitions quant à une perspective de gestion de données structurées et sémantiquement riches. D'autre part, ces mêmes systèmes sont limités aux applications dans lesquels les objets sont décrits par leurs noms et ne peuvent pas du coup tenir compte des liens complexes pouvant exister entre pairs.

2.4 Bases de données distribuées

2.4.1 Généralités

Dans le but d'organiser les données dispersées sur différents sites, le concept de base de données distribuée a été introduit. Une base de données distribuée est définie comme une collection multiple de bases de données logiquement liées et distribuées sur un réseau d'ordinateurs[60]. Un système de base de données distribuée (*Distributed Data management System*-DDMS) est le logiciel qui permet la gestion de la base de données distribuée. Son rôle principal est de rendre la distribution transparente à l'utilisateur en masquant les détails fastidieux liés à la gestion des données. Contrairement au Web et aux systèmes P2P où les données ne sont pas structurées et manquent de sémantique, les bases de données distribuées

mettent l'accent sur des données structurées ajoutant ainsi une sémantique supplémentaire. En effet, plus les données sont structurées, plus elles véhiculent une sémantique, plus aussi le langage d'interrogation de ces mêmes données doit être expressif. La sémantique de chacune de ces bases de données est en général capturée par un *schéma*. Certes, la structuration des données permet d'en comprendre plus le sens mais elle introduit un autre problème connu sous le nom d'*hétérogénéité de schéma*. En effet, des bases de données d'un domaine commun appartenant à différents utilisateurs sont la plupart du temps décrites de façon différente, c'est-à-dire par des schémas différents. L'interrogation d'un ensemble de sources de données hétérogènes s'en trouve plus difficile puisqu'une requête exprimée sur un schéma ne pourra pas forcément l'être sur un autre schéma de l'ensemble, étant donné la différence de vocabulaire. Cette hétérogénéité est résolue grâce à un processus appelé *médiation sémantique* et qui a pour but de détecter les éléments similaires de part et d'autre des schémas pour en unifier les vocabulaires en vue du partage de données (cf. section 2.4.3).

Un système de gestion de base de données distribuée, dans sa forme la plus simple, est constitué d'un serveur central. Ce dernier offre les services d'une base de données distribuée et s'occupe entre autres, de façon transparente, de la localisation des sources, de la décomposition des requêtes, de leur envoi aux sources, de la recomposition des résultats mais aussi de la gestion de la consistance des données. Cette approche fournit ainsi un support pour la gestion des schémas, l'expression de requêtes de haut niveau, le traitement et l'optimisation automatique de requêtes.

Un système de gestion de base de données distribuée est donc un logiciel complexe et les principes du génie logiciel doivent être appliqués pour aboutir à un système robuste et efficace. Pour les grands systèmes, la modularité est la clé pour une bonne architecture logicielle, avec une décomposition du système en composantes encapsulées avec des interfaces bien définies. Une telle modularité a été réalisée très tôt dans l'évolution des bases de données relationnelles, dans le système R[8]. La séparation des aspects proposée dans ce système est (avec des modifications) encore utilisée dans la plupart des produits de bases de données et aussi dans l'approche distribuée. Nous n'évoquerons pas les questions liées aux modifications de la base de données (gestion des transactions, connection, reprise, etc.) puisqu'elles vont au delà de notre travail. Cependant, le traitement des requêtes peut également servir de modèle dans notre contexte et il serait important de le voir en détail.

2.4.2 Traitement des requêtes

Le processus de traitement de requête est pris en charge par un processeur de requêtes, dont la structure est décrite dans la figure 2.6, et qui s'occupe d'interroger le système pour l'utilisateur de façon transparente. Les principales phases du traitement d'une requête sont le pré-traitement, la génération du plan de la requête et l'exécution de ce plan[60].

Le processeur prend en entrée une requête, exprimée sur le schéma central, et produit, dans un premier temps, un plan de requête spécifiant précisément comment la requête va être exécutée. La production de ce plan logique est précédée d'une phase de test de validité de la requête (bien-formation syntaxique, existence des sources d'informations mentionnées dans la requête, conformité des types d'attributs aux opérateurs utilisés). Généralement, ces plans sont représentés sous forme d'arbres ayant comme nœuds les opérateurs de la requête et comme feuilles les sources d'informations (par exemple les tables) qui doivent être combinées en vue d'obtenir le résultat final. Dans la première phase, la requête est mise sous forme algébrique pour un traitement ultérieur[43].

Une *optimisation logique* se charge d'effectuer des transformations algébriques incluant la normalisation, l'élimination des redondances et la simplification des expressions algébriques. L'étape suivante, appelée *localisation de données*, modifie le plan de requête avec une annotation indiquant pour chaque

relation de la requête les sources de données la supportant. Les phases de pré-traitement et d'optimisation sont toutes les deux basées sur les méta-données disponibles à propos des sources d'informations, principalement les schémas. Toutes les méta-données sont stockées dans une base de données (logiquement) séparée, le *catalogue*. Dans le cas distribué, il n'est pas nécessaire de maintenir un catalogue complet sur le site ; si l'information disponible est partielle, un plan de requête partiel peut être crée et envoyé aux autres nœuds, dans lesquels il sera raffiné jusqu'à l'obtention d'un plan complet. Cette requête annotée est ensuite passée au *module d'optimisation physique* qui, en utilisant des heuristiques ou un modèle de coût, génère un plan de requête optimisé tenant en compte les coûts de communication (bande passante, latence) entre sources mais aussi ceux de traitement à l'intérieur de chaque source[60].

Étant donné que l'énumération des plans alternatifs est exponentielle, une recherche exhaustive dans la totalité de l'espace de recherche des plans n'est pas souhaitable[86]. Par conséquent, de nouvelles heuristiques ont été introduites pour aider à la sélection des plans prometteurs. En définitive, plus de compromis doivent être faits entre la minimisation du temps de réponse et celui de l'effort de traitement de la requête. Le plan optimisé est finalement envoyé au *moteur d'exécution* qui a la charge de faire suivre les sous-plans aux sources appropriées, mais aussi de superviser leur évaluation. Le moteur d'exécution a aussi la responsabilité de déployer les canaux de communication entre les sources de données distantes contribuant au plan de la requête, mais aussi d'adapter ces plans durant l'exécution, en cas de problème[100].

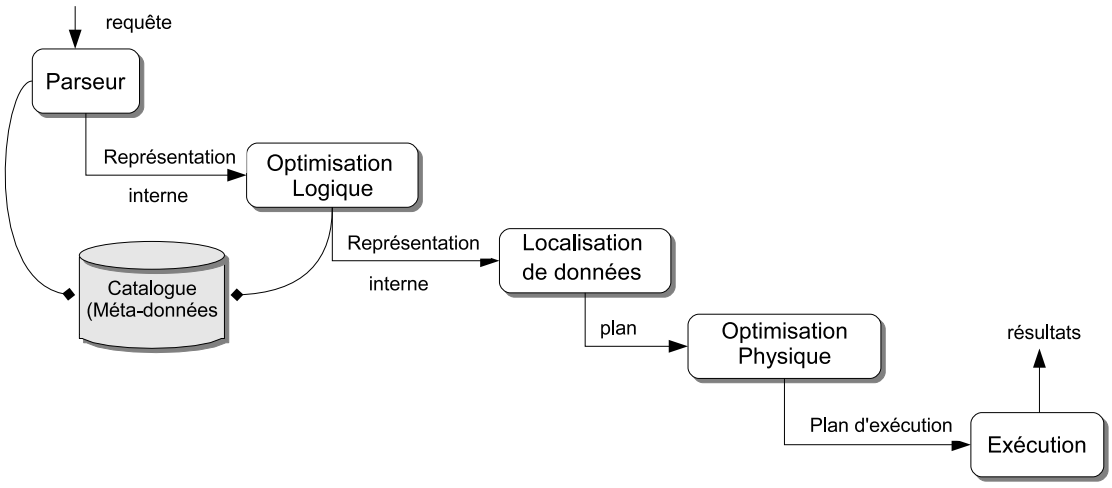


Figure 2.1 – Phases du traitement de requêtes dans un DDMS.

Bases de données fédérées. Souvent, les sources de données dans un contexte distribué sont autonomes et ne souhaitent pas être intégrées totalement de telle sorte que les plans de requêtes distribuées soient possibles. Un tel scénario est appelé base de données fédérée. Dans ce contexte, la structure coordonnatrice maintient un *schéma fédérateur* ou *schéma médiateur* et un ensemble de correspondances sémantiques avec les schémas participant à la fédération. L'interopérabilité des sources de données passe par un processus appelé *médiation sémantique* et permettant de découvrir les correspondances sémantiques entre le schéma médiateur et les différentes sources. Ces correspondances sémantiques sont nécessaires pour réécrire une requête sur le schéma médiateur en des requêtes sur les schémas sources. La découverte des correspondances se fait parfois en explorant aussi bien les connaissances linguistiques que celles structurelles véhiculées par les schémas. Cependant, par opposition aux systèmes étroitement cou-

plés, le système de gestion de la base de données fédérée doit convertir les plans de requêtes partiels en des plans de requêtes adaptés aux bases de données respectives. Finalement, les requêtes résultantes sont envoyées à travers les interfaces de requêtes offertes. Dans le but de réaliser une architecture flexible pour le système, la réécriture des bribes de plans de requêtes en des requêtes spécifiques aux sources est déléguée à des adaptateurs. Ceci permet une extensibilité du système au cas où de nouveaux types de sources doivent être intégrés.

2.4.3 Médiation sémantique

2.4.3.1 Le problème de la médiation sémantique

La médiation sémantique permet de venir à bout de l'hétérogénéité sémantique des schémas. Elle passe par l'établissement de correspondances sémantiques entres ces schémas, correspondances qui seront utilisées plus tard pour la réécriture d'une requête exprimée sur un schéma *source* en une autre requête, cette fois exprimée sur un schéma *cible*. Formellement, une correspondance sémantique décrit les relations entre les éléments de deux ou plusieurs schémas dans le but du partage et de l'intégration de données[72]. Ces correspondances sont spécifiées manuellement, cependant plusieurs approches ayant pour but d'automatiser ces tâches ont été proposées[92].

Les systèmes d'informations qui fournissent une interopérabilité entre plusieurs bases de données ont été appelés systèmes multi-bases [49], bases de données fédérées[44] et plus généralement systèmes de gestion de bases de données distribuées hétérogènes. Un mécanisme pour réaliser cet objectif est la construction d'un schéma médiateur réconciliant toutes ou certaines parties des schémas composantes. Ce schéma médiateur fournit un schéma global unifié pour les données du système. Les utilisateurs formulent leurs requêtes en termes du schéma médiateur. Par la suite, les correspondances sémantiques établies entre le schéma médiateur et les schémas des sources permettent de réécrire la requête originale en sous-requêtes exécutables sur les sources[115]. Par conséquent, bien qu'aucune donnée ne soit stockée au niveau du schéma médiateur (juste des méta-données), ce dernier est interrogé comme s'il détenait les données. Des adaptateurs localisés au niveau des sources sont utilisés dans le but de fournir des services de translation entre le schéma médiateur, le langage de requête et le modèle de donné, ou toute autre caractéristique locale incompatible avec le schéma médiateur. D'autre part, un schéma médiateur peut être construit sur la base d'autres schémas médiateurs, induisant ainsi une topologie sémantique appelée *arbre sémantique* et dans laquelle le schéma d'un nœud de l'arbre est réconcilié par son parent.

Les stratégies pour la définition des correspondances sémantiques entre le schéma médiateur et les schémas locaux ont été appelées : *global-as-view* (GAV), *local-as-view* (LAV), et *global-local-as-view* (GLAV)[63].

Global-as-View (GAV). Dans l'approche GAV, chaque entité dans le schéma médiateur est définie comme une vue sur les différents schémas sources à intégrer. Un avantage majeur de cette approche est que, répondre à une requête est assez triviale en faisant référence au schéma global. Cela veut dire que les requêtes reçues peuvent être facilement réécrites avec les termes utilisés par chaque source locale. Comme illustré dans la figure 2.2 un schéma global $G(A, R.B, C, S.B)$ est généré en résumant les schéma sources R et S. Tous les éléments dans les schémas sources ont des noms correspondants dans le schéma global même si quelques uns d'entre eux, tels que $R.B$ et $S.B$, partagent le même sens. Cependant, il devient difficile de mettre à jour le schéma global à cause de la dépendance entre le schéma global et les schémas locaux. Par exemple, si le schéma global a été mis à jour (par exemple de nouveaux éléments ont été ajoutés) tous les schémas sources doivent mettre à jour leur vue locale sur le schéma global. D'autre part, l'ajout ou la suppression

de sources peut résulter en des modifications considérables sur le schéma global. Comme illustré dans la figure 2.2, si un nouveau nœud T a été ajouté au système, le schéma global doit être modifié en $G'(A, R.B, C, S.B, T.A, D)$.

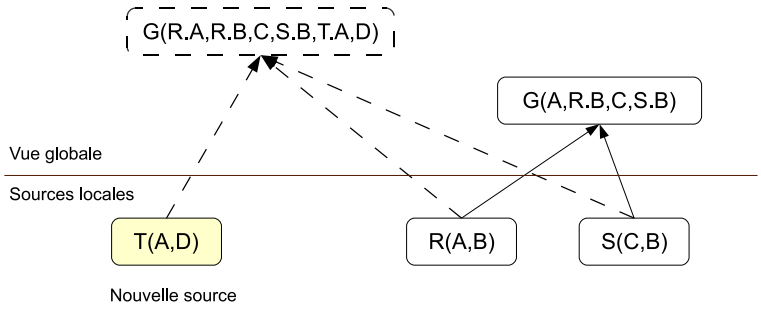


Figure 2.2 – Un exemple GAV.

Local-as-View (LAV) Contrairement à l'approche GAV, dans l'approche LAV les vues sur les sources sont utilisées dans le sens opposé. Ces vues définissent comment l'information locale est liée au schéma global en exprimant une correspondance entre chaque relation dans le schéma local et une (des) relation(s) dans le schéma global[63]. Un exemple LAV est illustré dans la figure 2.3, en comparaison à l'approche GAV. Le principal avantage de l'approche LAV par rapport à GAV est qu'il n'y a pas de dépendance par rapport au schéma global. Dans l'approche LAV, chaque schéma source est rattaché au schéma global grâce à des correspondances. L'ajout de nouvelles sources au système nécessite seulement la définition des correspondances nécessaires entre le schéma source et le schéma global. Cependant, dans cette approche, répondre à une requête devient plus difficile parce que la reformulation de requête est difficile à réaliser.

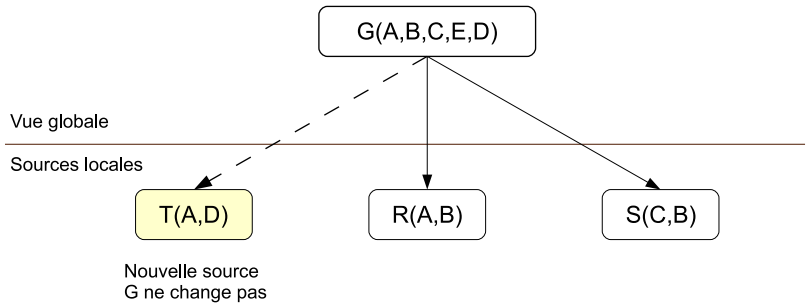


Figure 2.3 – Un exemple LAV.

Gloabl-and-Local-as-View (GLAV). Dans l'approche GLAV, l'intégration entre le schéma médiateur et les schémas locaux est réalisée en combinant les pouvoirs d'expression des approches GAV et LAV[39]. Dans l'approche GLAV, l'indépendance du schéma global, la maintenance nécessaire pour ajouter une nouvelle source et la complexité de la reformulation des requêtes sont les mêmes que dans l'approche LAV. Cependant, GLAV peut créer une vue sur les sources en générant une vue sur le schéma global décrite par les descriptions des sources (c.f. figure 2.4). Par conséquent, GLAV peut dériver des données en utilisant les vues sur les schémas sources, ce qui est plus expressif que LAV. D'autre part, il permet la reformulation sur le schéma global, ce qui va au

delà du pouvoir d'expression de GAV. On peut remarquer que G' dans la figure 2.4 est juste la conjonction de G et du schéma du nouveau nœud T.

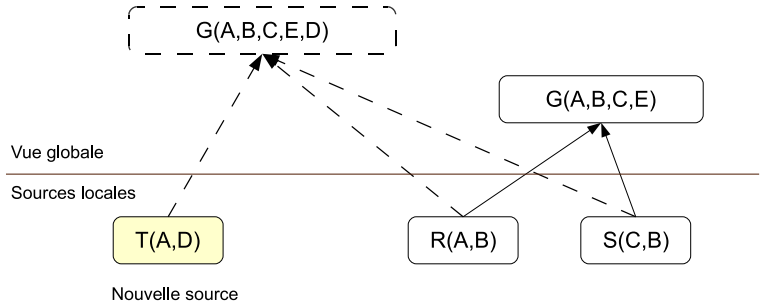


Figure 2.4 – Un exemple GLAV.

Approche	Réécriture de requête	mise-à-jour source
GAV	facile	difficile
LAV	difficile	facile
GLAV	difficile	facile

Table 2.1 – Comparaison des approches GAV, LAV et GLAV

2.4.3.2 Création et maintenance des correspondances

Le problème de la création et de la maintenance des correspondances sémantiques a déjà été étudié dans un contexte autre que le Pair-à-Pair, notamment dans les systèmes d'intégration de données. On peut trouver dans [92] un résumé exhaustif des approches d'intégration de schémas, ainsi que les comparaisons entre plusieurs méthodologies. Les auteurs ont suggéré les critères de classification orthogonales ci-dessous :

- *Schéma/Instance*. Les approches basées sur les schémas ne considèrent que les informations sur les schémas et pas les instances de données[13][87]. Les informations sur les schémas incluent les noms des éléments, leurs descriptions, leurs relations, les contraintes, etc. Les approches basées sur les instances utilisent des méta-données et des statistiques collectées à partir des instances de données pour annoter les schémas[76], ou trouvent directement les éléments de schémas liés, par exemple en utilisant l'apprentissage automatique[32].
- *Elément/Structure.* Un appariement au niveau schéma détermine les correspondances entre les éléments de schémas pris individuellement, par exemple entre les attributs[66]. L'approche structurelle compare des combinaisons d'éléments qui apparaissent simultanément dans les schémas, par exemple des classes ou des tables dont les ensembles d'attributs correspondent approximativement[13].
- *Linguistique/Contrainte*. L'approche linguistique utilise les noms des éléments des schémas et leurs descriptions textuelles. La découverte des correspondances est faite en vérifiant l'égalité des noms ou en comparant leurs ensembles de synonymes et d'hyperonymes par le biais de thésaurus spécifiques au domaine. La méthode basée sur les contraintes utilise les types de données, les domaines de valeurs, l'unicité, les cardinalités, etc. Les contraintes entre les éléments des schémas telles que les intégrités référencielles peuvent aussi être utilisées.

- *Cardinalité des correspondances*. Les approches diffèrent par la cardinalité des correspondances qu'elles établissent. Certaines approches produisent des correspondances *un-à-un* (1:1) tandis que d'autres établissent des correspondances *un-à-plusieurs* (1:n), comme par exemple celle qui permet la décomposition de $PrenomNom$ dans un schéma source en $Prenom$ et Nom dans un schéma cible.
- *Informations auxiliaires*. Les méthodes diffèrent aussi par le fait qu'elles autorisent ou pas l'utilisation d'informations auxiliaires telles que des dictionnaires, des thésaurus, des informations sur des correspondances antérieures ou encore des indications de l'utilisateur.
- *Combinaison d'approches*. Il est possible de combiner les approches en utilisant plusieurs des critères listés ci-dessus. Les approches *hybrides*[13][66][78] utilisent plusieurs critères à la fois, tandis que les approches *multiples* appliquent indépendamment les algorithmes de mise en correspondance sur les deux schémas et combinent par la suite les résultats obtenus[32].

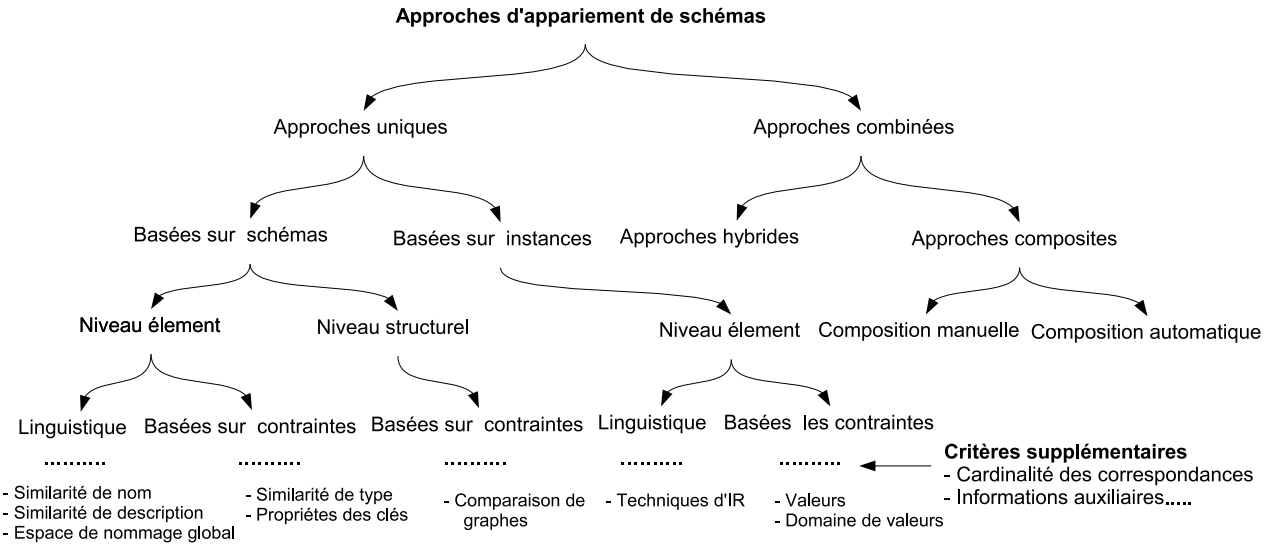


Figure 2.5 – Taxonomie des techniques de découverte de correspondances sémantiques entre schémas.

2.4.4 Quelques projets de recherches

Dans les années 90, plusieurs projets de recherches ont commencé à mettre en place des solutions pour interconnecter des sources de données via le Web. TSIMMIS[89] est l'un de ces premiers projets pour l'intégration de sources de données semi-structurées à partir du Web. Il utilise une approche médiateur-adaptateur basée sur GAV et dans laquelle un médiateur central prend en charge la génération des plans de requêtes tandis que des adaptateurs spécialisés réécrivent des requêtes (partielles) en des requêtes spécifiques aux sources. TSIMMIS se focalise sur la sélection et la projection en se basant sur l'argument selon lequel les jointures sont peu probables dans le contexte du Web. Les autres systèmes d'intégration de données sont Information Manifold[65], AMOSII[53], Garlic[54], DISCO[110] et Strudel[37](le prédécesseur de Piazza, cf. section 2.6.6)

Systèmes de gestion de bases de données distribuées à grande échelle. La plupart des approches de bases de données distribuées ont pour but l'intégration de quelques sources de données. Deux ap-

proches plus récentes traitent du problème de l'interconnection entre un nombre plus grand de nœuds, Mariposa[107] et ObjectGlobe[22].

Mariposa utilise un modèle micro-économique pour l'optimisation du coût dans le processus de génération de plans. Chaque client alloue un budget à sa requête, et un générateur de requête centralisé demande aux sources de données leurs offres concernant l'exécution des opérateurs de la requête. Le fournisseur le moins coûteux est sélectionné. Cette approche favorise le passage à l'échelle du système, comparé à l'estimation de coût purement centralisée. Cependant cette approche n'est pas facilement applicable aux réseaux ouverts car la solution à la gestion des devises n'a pas été abordée[107].

Le but de ObjectGlobe[22] est d'accomplir la vision de Internet comme une base de données globale. Il peut être vu comme un processeur de requêtes distribuées dans lequel un plan de requête est conçu et exécuté en se basant sur les informations des schémas. ObjectGlobe suppose un catalogue global dans lequel toutes les informations sur les schémas sont stockées. Chaque nœud peut joindre ou quitter le réseau dynamiquement. Toutefois, il doit se faire enregistrer a niveau du catalogue central. Contrairement aux systèmes traditionnels basés sur les médiateurs, il n'y a pas que les sous-requêtes qui sont déléguées aux fournisseurs de données, mais aussi toute l'exécution de la requête est distribuée. Ainsi donc, chaque nœud peut fournir aussi bien des données qu'une capacité de calcul, et le plan de requête est conçu pour optimiser les performances à travers tous les nœuds.

2.4.5 Limites des systèmes de gestion de bases de données distribuées

Les systèmes de gestion de bases de données distribuées offrent des capacités avancées de gestion de données telles que la gestion de schémas, des langages de requêtes de haut niveau, le contrôle d'accès, le traitement et l'optimisation automatique de requêtes, etc. L'utilisateur peut ainsi accéder de façon transparente à des données localisées dans plusieurs sources. Cependant, ces systèmes, bien qu'ayant amélioré le passage à l'échelle du traitement de requêtes distribuées, sont encore loin de passer à l'échelle de l'Internet. Par exemple Mariposa a pour but l'intégration de « 1000 sites (machines) ou plus ». Les évaluations réalisées dans les autres systèmes concernent quelques dizaines de sites. En effet, le contrôle central en général exercé par un seul serveur constitue un obstacle pour le passage à l'échelle. D'autre part, avec la conception d'un schéma central médiateur qui, jusque là, facilitait l'intégration, les sources de données ne peuvent pas changer de façon significative sans risquer de violer les correspondances avec le schéma réconciliateur. La conséquence est que les nœuds ont une faible autonomie, ce qui viole une des exigences fondamentales d'un système de partage de données. De même, de nouveaux concepts ne peuvent être ajoutés au schéma médiateur que par l'administrateur central. Ce qui fait que la conception du schéma central est difficile, sa modification aussi. Cette approche centralisée présente des limites car n'étant plus adaptée à l'extensibilité d'un environnement regroupant plusieurs sources de données en mutation telle que le Web par exemple.

Dès lors, il s'avère nécessaire de développer de nouveaux outils de partage de données préservant non seulement la sémantique de ces dernières, mais permettant en plus de les interroger avec des langages de requêtes riches. En outre, ces mêmes outils devraient permettre le partage décentralisé de données et la définition de relations sémantiques entre sources sans passer par un schéma médiateur. Chaque source devra donc pouvoir ajouter de nouvelles données, définir de nouveaux schémas et les lier à ceux qui existent déjà, sans porter atteinte à la cohérence globale du système. Ceci constitue un avantage notoire pour l'extensibilité du système. Les systèmes pair-à-pair semblent bien adaptés à ce type d'application de partage décentralisé de données et les bases de données distribuées constituent un bon point de départ pour la plupart des PDMS actuels.

2.5 Les réseaux Pair-à-Pair

2.5.1 Généralités

Un réseau Pair-à-Pair est déterminé par trois caractéristiques essentielles, auto-organisation, communication symétrique et contrôle distribué[95] :

- **Auto-Organisation.** Par opposition aux autres systèmes distribués, les réseaux P2P ne nécessitent pas d'administration et la maintenance nécessaire est prise en compte par les algorithmes sous-jacents. Ces réseaux autorisent l'arrivée de nouveaux pairs ou le départ de pairs courants tout en adaptant les connections réseaux en conséquence.
- **Communication Symétrique**. Par opposition au modèle client/serveur dans lequel chaque nœud joue le rôle de client ou de serveur, dans un réseau P2P chaque nœud peut se comporter à la fois comme client et serveur.
- **Contrôle Distribué**. Les systèmes P2P ne disposent pas de point de contrôle central. Le contrôle est distribué entre les pairs de telle sorte qu'aucun pair ne devienne un goulot d'étranglement. En cas d'indisponibilité d'un pair, sa part de contrôle est prise en charge par les autres nœuds.

Les systèmes P2P qui ne présentent pas toutes ces caractéristiques en même temps sont appelés *systèmes P2P hybrides*.

	Système P2P	Client/Serveur
Coût	moins cher + maintenance facile	plus cher+maintenance difficile
Indépendance	élevée	faible
Flexibilité	élevée	faible
Passage à l'échelle	élevée	limité
Fiabilité	moins fiable (PC personnels)	plus fiable (serveur dédié)
Sécurité	limitée	élevée
Robustesse	robuste	modérée
Communication	symétrique	asymétrique

Table 2.2 – Avantages et inconvénients des systèmes P2P (comparés au modèle Client/Serveur)

La première génération de réseaux P2P est celle appelée *non-structurée*. Dans cette catégorie un nouveau pair joint le réseau en se connectant au hasard aux pairs faisant déjà partie du réseau, ce qui aboutit à un réseau avec une topologie quelconque. Gnutella[91] constitue le système le plus représentatif dans ce domaine. Dans ces systèmes, les requêtes sont diffusées de la même façon et sans tenir compte de leurs contenus, récursivement à tous les pairs en ligne jusqu'à ce qu'un certain horizon, défini par l'initiateur de la requête, soit atteint. Ainsi donc, les objets stockés au delà de cet horizon ne sont pas atteints.

La génération suivante a été appelée réseaux P2P *structurés*. Cette étape a vu le développement de systèmes basés sur les *Tables de Hachage Distribuées* (DHT). Chaque objet fournit par un pair est associé à une clé et la seule opération de recherche possible est la recherche par clé. Cette restriction est compensée par l'avantage suivant des DHTs : chaque objet dans le réseau correspondant à la clé de recherche est sûr d'être trouvé, si on ne tient pas en compte les fautes généralisées à l'échelle du réseau. Le routage de requêtes est donc très efficace et les nœuds sont organisés souvent sous forme de graphes réguliers.

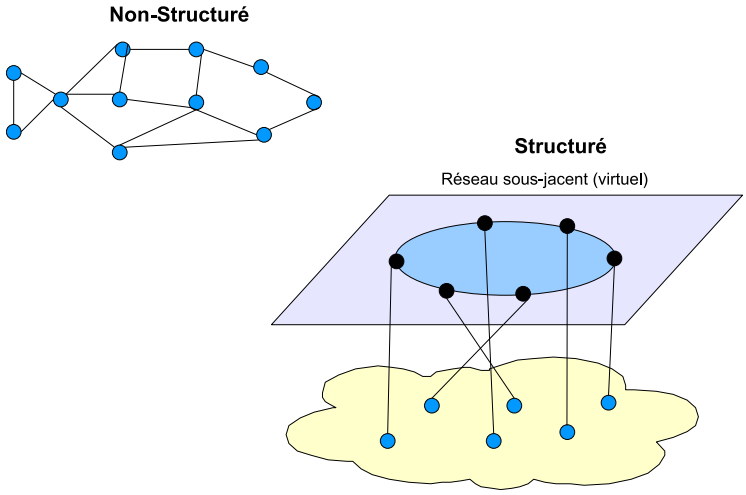


Figure 2.6 – Différentes approches P2P

Cependant cette classification en *structuré* et *non-structuré* peut être revue en tenant compte des considérations suivantes[83] :

- Certaines approches récentes[75][102] ne peuvent pas être catégorisées en *structurée* ni en *non-structurée*. En effet, elles utilisent des topologies formées au hasard et qui peuvent cependant converger, au fil du temps, vers un graphe avec des caractéristique régulières.
- Certes, la distribution des requêtes par inondation à tout le réseau ne requiert pas une topologie sous jacente régulière, cependant elle pourrait en bénéficier. Ceci montre que le routage de requête et la topologie du réseau devraient être vus sont des aspects différents au niveau conceptuel, bien que liés.
- Il a été établi que dans certains cas une combinaison DHT/Inondation peut surpasser l'inondation simple aussi bien que le DHT pur[67].

Par conséquent, une meilleure approche pour classifier les réseaux P2P serait celle en accord avec leurs méthodes d'indexation et non leur topologie comme énoncé dans [95]. Ceci conduit à trois méthodes d' indexation que nous décrivons dans la suite : *locale*, *centralisée* et *distribuée*.

2.5.2 Indexation dans les réseaux P2P

2.5.2.1 Indexation locale

Dans ce type d'indexation, il n'existe pas d'index distribué et le routage de requêtes est fait par inondation[91][116] ou aléatoirement[70][4]. Dans le cas de l'inondation, la requête est copiée durant la propagation et ses copies envoyées en parallèle, tandis que dans le cas de l'approche aléatoire la requête est passée de pair en pair jusqu'à la découverte des données. Notons qu'aucun système utilisant ces approches n'a encore été déployé à grande échelle. Cependant, plusieurs tentatives ont été faites pour améliorer ces techniques de routage par le biais d'heuristiques[116] ou en préférant les nœuds ayant plus de connections[4]. Toutefois, il est d'un commun accord qu'un index purement local ne passe pas à l'échelle[96].

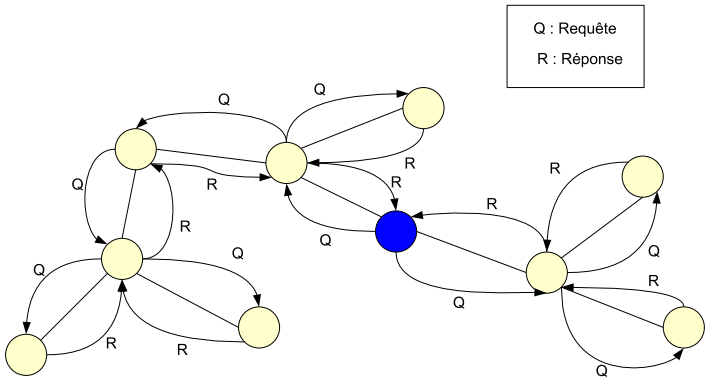


Figure 2.7 – Indexation Locale.

2.5.2.2 *Indexation centralisée*

L'idée d'un index centralisé est contraire à l'une des caractéristiques principales d'un système P2P qui élimine tout contrôle central à cause des lourdeurs imposées par la charge sur le nœud central. Cependant, Napster a montré qu'un nœud central peut supporter cette charge dans le cas où seulement l'accès à l'index est centralisé et les données partagées ne transitent pas par le centre[81][101]. Les recherches dans ce domaine n'ont pas été poursuivies, probablement à cause de l'index central qui va dans le sens contraire de l'esprit du P2P. Cependant, l'idée de séparer la gestion de l'index de celle des données a été explorée et elle a conduit à la mise en place des systèmes super-pairs.

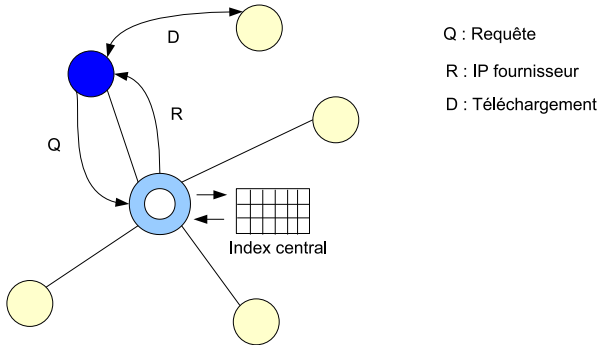


Figure 2.8 – Indexation centralisée.

2.5.2.3 *Indexation distribuée*

Cette classe contient la plupart des systèmes P2P. Aucun nœud n'a une connaissance totale du réseau, au contraire, chaque nœud détient un index partiel et suffisant pour lui permettre de diriger les requêtes vers les pairs pertinents pour la requête en accord avec leurs contenus.

Réseaux avec Raccourcis (*Shortcut Networks*). Les réseaux avec raccourcis permettent une topologie adaptative en changeant les connections en fonction des requêtes et des statistiques sur les réponses[108][104]. Par conséquent, chaque pair essaie de minimiser la distance avec les pairs

qu'il sollicite le plus souvent en créant de nouveaux raccourcis au lieu de faire des détours dans le réseau.

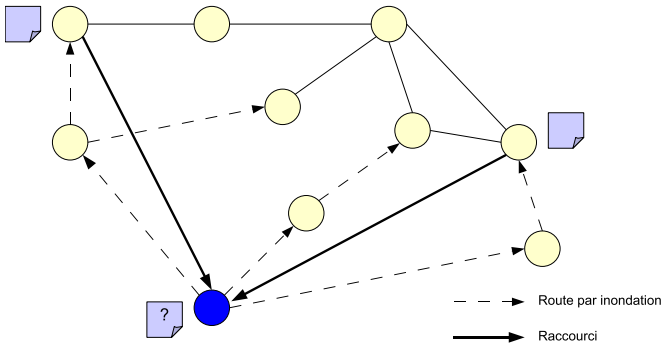


Figure 2.9 – Réseaux avec Raccourcis.

Inondation avec filtrage Les DHTs ont été conçues pour la recherche basée sur les clés et donc les autres types de recherche avec des requêtes complexes ne peuvent pas être supportés aussi facilement par les DHTs. Un nombre important de systèmes supportant des requêtes complexes utilisent l'inondation pour distribuer les requêtes tout en minimisant cependant les destinations en se basant sur la requête et sur l'index distribué. Cet index qui sert de filtre pour le réseau peut être conçu à l'avance ou en se basant sur les statistiques sur les requêtes. Dans le premier cas, les pairs envoient des résumés de leur contenus qui sont stockés dans les index et qui seront plus tard mis en correspondance avec les requêtes[18][41][59][83]. Dès lors, une requête n'est transmise à un pair que si la description du contenu de ce dernier correspond à la requête. Dans le second cas, les requêtes sont d'abord envoyées et les indexes construits au fur et à mesure en fonction des réponses des pairs[9]. Ainsi donc, l'index est mis en place par le biais du processus de traitement de requêtes et donc aucun message de maintenance d'index n'est nécessaire.

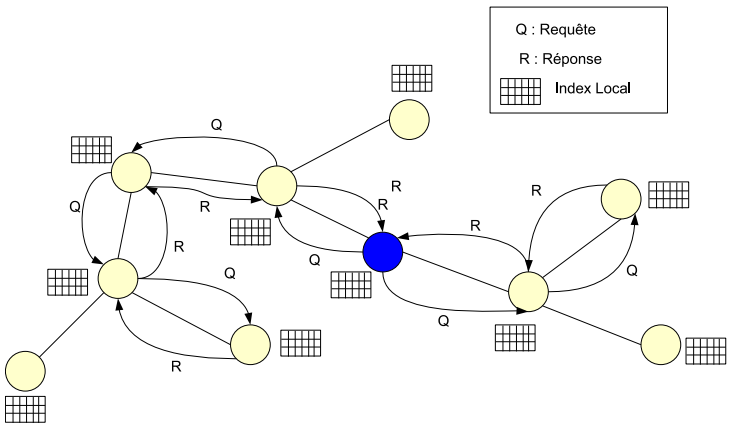


Figure 2.10 – Inondation avec filtrage.

Réseaux Super-pairs. La topologie super-pair profite de l'hétérogénéité des nœuds en termes de pouvoir de calcul, de capacité de stockage, de bande passante, de disponibilité, etc. Ainsi, en tenant compte des capacités des pairs, les pairs les plus disponibles et puissants, appelés *super-*

pairs, se voient affecter des tâches complexes tels que le routage de requêtes, la médiation de schémas[116]. Ces super-pairs forment un réseau entre eux et acceptent la connection de pairs simples et deviennent par là leurs points d'accès au réseau. Des systèmes de partage de fichiers tels que Gnutella[91] dans sa version 0.6 ou Fastrack [1] utilisent cette approche. Les super-pairs assument les responsabilités à la fois d'indexation et de traitement de requêtes. Le routage dans cette catégorie de réseau se fait le plus souvent en deux phases, une première phase dans laquelle les requêtes sont envoyées dans le réseau de super-pairs qui les distribuent ensuite aux pairs qu'ils administrent. A noter que le réseau super-pair peut être basé sur l'une des topologies précédemment citées. Le routage se fait en général grâce à un index[27] ou une table de routage[83]. L'approche super-pair est aussi utilisée dans SenPeer.

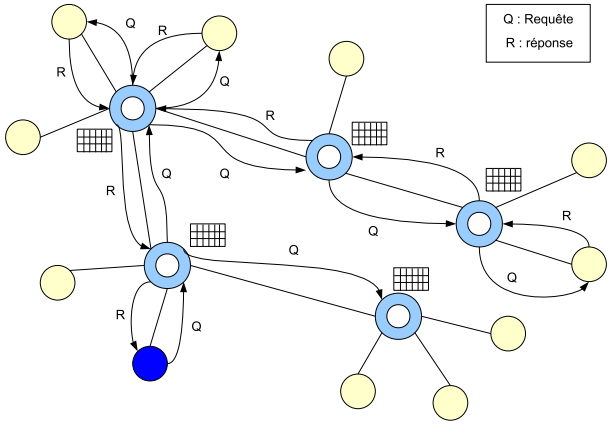


Figure 2.11 – Indexation avec Super-pair.

Tables de Hachage Distribuée. Les DHTs constituent la technique d'indexation distribuée la plus répandue. Chaque objet du réseau se voit assigner une clé et la seule opération de recherche possible est la recherche par clé. Le principe des DHTs est d'assigner chaque portion de l'espace des clés à un nœud et de créer ensuite une connection structurée entre tous les nœuds. Par conséquent, pour chaque clé il existe une seule route entre l'initiateur de la requête et le pair responsable de cette clé. Les DHTs permettent de réduire rapidement la quantité de nœuds nécessaires pour répondre à une requête sur des données répandues dans le système P2P. Étant donné une clé de taille n les données correspondantes peuvent être localisées de façon efficace en utilisant $O(\log n)$ messages réseau, avec n le nombre de pairs dans le réseau. Plusieurs protocoles implémentent des DHTs et fournissent des primitives de recherche pour faire correspondre une clé donnée à un pair spécifique.

La topologie dominante pour les DHTs est basée sur les arbres de Plaxton[90]. Les systèmes tels que Chord[106], Pastry[98] ou OceanStore[62] reposent sur cette structure. D'autres graphes, tels que les graphes de Bruijn[56][80] et Butterflies[74], sont utilisés pour les topologies P2P .

CAN[93] (*Content Addressable Network*) utilise une approche différente de celle ci-dessus. Les clés sont associées à des coordonnées dans un espace cartésien de dimension d, et chaque pair est responsable d'une zone de clés. Les pairs couvrent en permanence l'ensemble de l'espace des clés. Ainsi, lorsqu'un nouveau pair arrive, les zones sont partagées. Cela permet d'avoir une table de routage de taille fixe (par exemple $2d$, si on ne prend qu'un voisin dans chaque direction).

[1] Le protocole de FasTrack n'a pa été publié.

P-Grid[3] utilise une topologie dérivée de la structure de treillis dont le mécanisme de construction reste efficace. Comme les autres approches DHT, l'espace de clé est réparti entre les pairs. Dans ce cas-ci, la partition est organisée en accord avec les préfixes de clés : chaque nœud est responsable d'un préfixe spécifique.

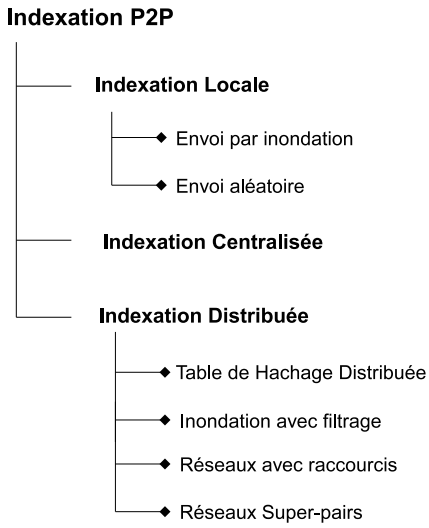


Figure 2.12 – Indexation dans les réseaux Pair-à-Pair.

2.6 Partage de données dans un PDMS

2.6.1 Définition d'un PDMS

Les PDMS sont une convergence naturelle entre les systèmes P2P et les bases de données distribuées. En effet, un PDMS peut être vu comme une généralisation d'un système d'intégration de données car étant un système dynamique et distribué dont le but principal est le partage autonome de données structurées à grande échelle. En observant l'évolution de la recherche dans les systèmes P2P et les bases de données, on se rend compte de cette convergence. En effet, les premiers efforts de recherche étaient concentrés sur les topologies et le routage de requêtes simples basées uniquement sur un attribut. Une seconde étape a été l'introduction de la recherche multi-dimensionnelle (avec plusieurs attributs) et, à l'état actuel, les travaux sont orientés sur des requêtes plus expressives. En accord avec le pouvoir d'expression des requêtes, les systèmes P2P peuvent être classés en systèmes basés sur les clés, les mots-clés ou encore les schémas[29]. Ce dernier type de système correspond aux PDMS et peut être vu comme une évolution des bases de données distribuées vers une plus grande distribution. Contrairement à un DDMS, un PDMS adopte une approche entièrement décentralisée pour le partage des données. Chaque pair maintient sa propre source de données, et par conséquent, son propre schéma, qu'il peut partager en totalité ou en partie avec le reste du système. En distribuant le stockage de données, le traitement et l'exécution des requêtes entre pairs, ces types de systèmes peuvent passer à l'échelle d'un nombre important de nœuds sans nécessiter un contrôle central ni de serveurs puissants. De plus, l'efficacité et la qualité de service sont fournies en dépit de l'apparition de fautes dues à la nature dynamique du système.

D'autre part, un PDMS hérite des caractéristiques essentielles d'un système P2P classique, c'est-à-dire, l'auto-organisation, la communication symétrique et le contrôle distribué, notamment pour les requêtes.

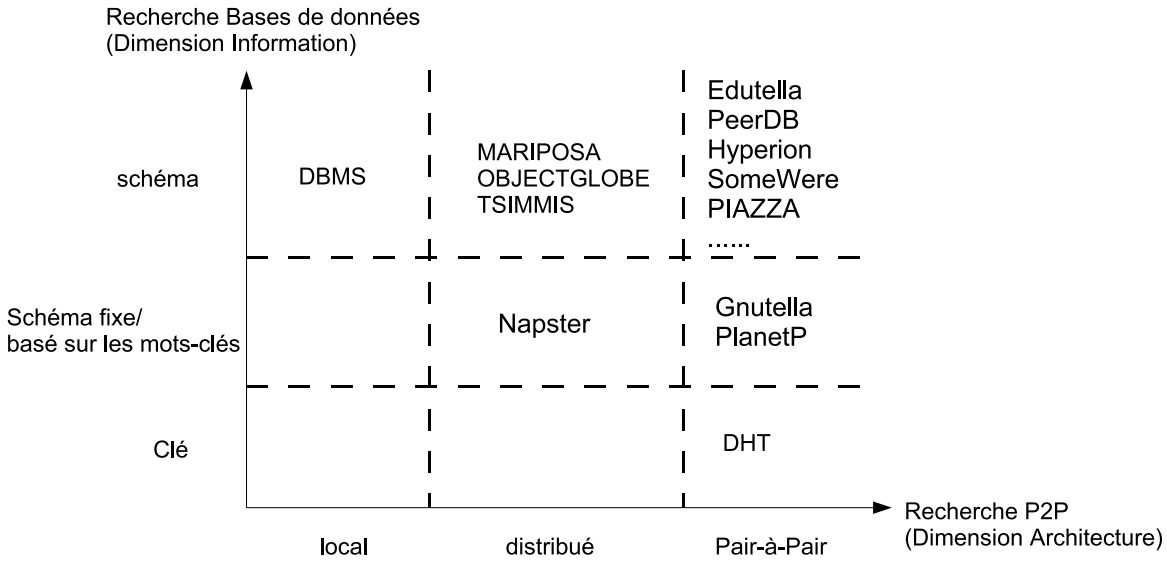


Figure 2.13 – Combinaison entre schémas et distribution[83].

L'autonomie est la caractéristique la plus significative d'un PDMS. Elle peut être décrite de trois points de vue[112]. L'autonomie dans le *stockage* fait référence à la liberté pour le pair de déterminer le contenu qu'il voudrait stocker, mais aussi du contenu qu'il voudrait partager avec les autres pairs. L'autonomie dans l'*exécution* signifie la possibilité pour chaque pair de traiter localement et de répondre à une requête tout en autorisant la coordination avec les autres pairs participant à la réponse pour effectuer de façon efficace le traitement de la requête distribuée. L'autonomie dans la *durée de vie* fait référence à la liberté pour le pair de joindre ou quitter le réseau à tout moment, ce qui diffère fortement d'avec les bases de données distribuées. Enfin, l'autonomie de *connection* concerne la topologie du système et autorise un pair à choisir à quels pairs mais aussi à combien d'entre eux se connecter.

Comme nous l'avons vu dans les sections précédentes, un certain nombre d'algorithmes, toutes appartenant à la famille DHT, ont été définis pour la recherche unidimensionnelle (avec un attribut). Cependant, cette recherche devient beaucoup plus compliquée dans le cas d'une requête complexe impliquant plusieurs attributs. Ces approches sont difficilement applicables dans le cas du PDMS parce que cela nécessite la mise en place d'un réseau sous-jacent pour chaque attribut, ce qui a un coût prohibitif.

D'autre part, un certain nombre de suppositions faites par les DDMS durant les phases de découverte de correspondances et de traitement de requêtes sont révisées par les PDMS en ajoutant des exigences additionnelles :

- *Structure centrale* : Là où un DDMS n'autorise que les correspondances sémantiques les sources et le schéma médiateur, un PDMS permet d'établir ces correspondances entre n'importe quel couple de sources de données et ne supporte pas de schéma global. En l'absence de schéma global, les pairs doivent pouvoir exprimer les requêtes sur leurs propres schémas. Dans ce cas, le PDMS reformule la requête sur les voisins du pair en utilisant les correspondances sémantiques. D'autre part, chaque pair pourra accepter ou refuser l'exécution d'une requête en fonction de sa charge de travail, ce qui n'est pas possible dans les DDMS puisque la sélection de ces pairs est faite de façon

centralisée par le module d'optimisation de requêtes.

- *Uniformité* : L'optimisation traditionnelle dans les DDMS suppose que toutes les puissances de calcul et les connections ont respectivement la même valeur et la même vitesse. De plus chaque pair est supposé pouvoir prendre en charge l'exécution de n'importe quel opérateur algébrique. Ces suppositions deviennent caduques dans un PDMS. En effet, les PDMS profitent de l'hétérogénéité des pairs en termes de capacité de stockage, de puissance de calcul et de bande passante pour assigner plus de tâches à certains pairs.
- *Plans de requêtes statiques* : Contrairement aux DDMS dans lesquels les plans de requêtes sont statiques, l'adaptabilité des plans de requêtes aux départs des pairs durant l'exécution est une caractéristique principale des PDMS. L'adaptabilité peut être réalisée en changeant entièrement ou partiellement le plan d'exécution courant.

Dans les DDMS, un seul pair est responsable de toutes les étapes du processus de traitement d'une requête à l'exception de la dernière, étant donné que plusieurs pairs peuvent participer à l'exécution de la requête. A l'opposé, les PDMS permettent à chaque pair de pouvoir assister à toutes les étapes du processus de traitement des requêtes.

Il existe un ensemble de PDMS proposés dans la littérature et évoquant le processus de médiation sémantique et de traitement de requêtes. Cependant, il semble surréaliste de réaliser une approche qui soit optimale pour tous ces aspects. Au contraire, des décisions doivent être prises et doivent aller dans le sens de compromis entre la médiation sémantique, l'expressivité des requêtes et l'efficacité du processus de traitement des requêtes.

Notre approche est complémentaire à ces propositions antérieures. Nous nous intéressons particulièrement à la médiation de données et au traitement des requêtes dans le cas ou chaque pair abrite des données, de n'importe quel domaine, conforme à un modèle de données quelconque et, par conséquent, dispose de son propre langage d'interrogation. Dans la suite, nous mettons l'accent sur les paramètres à prendre en compte dans la conception d'un PDMS et nous classons les systèmes existants en accord avec ces paramètres.

2.6.2 Stockage et Accès aux données

Le choix du mode de stockage des données et, par conséquent, de leur mode d'accès est important dans la conception d'un PDMS. Le modèle de données est étroitement lié au choix du langage d'interrogation.

Modèle de données. Plusieurs modèles de données ont étés proposés pour le stockage des données dans un PDMS. Nous ne pouvons pas les énumérer tous dans la suite mais nous en citons quelques différences. Les premiers modes de stockage, en particulier dans les systèmes P2P de partage de fichiers étaient basés sur des schémas fixes avec des attributs standard utilisés dans tout le système. Certes, cette approche présente des limitations mais permet de se départir du problème d'interopérabilité entre schémas. Comme système utilisant cette approche on peut citer le réseau structuré MAAN[24] avec modèle de données de tuples simple. Une approche plus complexe a été proposée avec l'arrivée du relationnel avec des systèmes tels que PeerDB[84] ou PIER[47][46]. Le format XML, de part sa vocation de format d'échange pour le Web, a aussi été utilisé dans plusieurs systèmes qui sont entre autres XPeer[103] et Piazza. Pour interconnecter des bases de connaissances distribuées, le modèle RDF a été adopté dans plusieurs systèmes tels ques RDFPeers[24], NeuroGrid[52] ou SQPeer[59]. Par ailleurs les ontologies sont de plus en plus introduites comme langage de schéma pour décrire des informations[18][38]. Des approches

comme [38] et [18] supportent entièrement des ontologies ou permettent le raisonnement distribué comme SomeWhere[5].

Langage de requêtes. Le langage d'interrogation va au delà de la recherche simple par mots-clés. Bien entendu, le choix du langage d'interrogation dépend du modèle de données : les systèmes relationnels utilisent un sous-ensemble de SQL, et les systèmes XML un sous-ensemble de XQuery. Quant aux approches basées sur RDF, elles utilisent des approches basées sur la comparaison de graphes. Ces langages permettent de formuler des requêtes complexes impliquant plusieurs attributs comme dans les bases de données distribuées. A notre connaissance, très peu de systèmes offrent la possibilité de faire cohabiter plusieurs langages d'interrogation dans le même PDMS. Les systèmes supportant l'expression de requêtes logiques dans le contexte du Web sémantique ajoutent aussi de l'expressivité dans l'interrogation. En effet, les utilisateurs peuvent peuvent introduire de nouveau termes en interrogeant le système.

2.6.3 Topologie et routage de requête

Comme nous l'avons vu dans la section 2.12, il existe différentes approches pour organiser les pairs en une topologie sous-jacente. Le routage de requêtes a la responsabilité de trouver les pairs pertinents pour la requête, en se basant sur les connaissances concernant les schémas des pairs. Ces connaissances sont distribuées entre les pairs de telle sorte que chaque pair dispose d'un sous-ensemble d'informations qui lui permettent de localiser les données pertinentes pour sa requête. Il est d'un commun accord que pour une recherche unidimensionnelle, les DHTs supplantent les autres solutions. Cependant, pour les PDMS avec des requêtes multi-attributs, le problème n'a pas encore été élucidé. Nous donnons dans la suite quelques approches selon la catégorisation faite dans la section 2.12.

DHT multi-dimensionnelle. L'idée de base consiste à créer des topologies sous-jacentes pour chaque attribut. Cette approche est utilisée dans les systèmes tels que PIER[46] et RDFPeers[24]. L'avantage est que tous les accès sont garantis avec un nombre de sauts de l'ordre de $O(\log n)$. Ceci permet une efficacité des requêtes tant que le nombre d'attributs concernés n'est pas élevé. En outre, la maintenance de la topologie et l'insertion des données sont coûteuses bien que dépendant seulement du nombre d'attributs.

Réseaux non-Structurés. Ces réseaux ne créent pas d'index pour tous les attributs. A la place, les requêtes sont routées vers des destinations prometteuses et sont mises en correspondance avec les schémas des pairs. On assiste à une inondation filtrée. Cette approche est utilisée dans Piazza[41] et dans Chatty Web[1].

Réseaux avec raccourcis. Ce type de réseau n'impose pas une structure régulière de graphe à la topologie. Les connections sont optimisées en créant des *raccourcis* dont l'établissement est basé sur l'observation des résultats des requêtes précédentes. Certes, la topologie n'est pas explicitement construite, mais l'ensemble des raccourcis aboutit à une structure de graphe spécifique. PeerDB[84], REMINDIN'[108] et INGA[68] peuvent être classés dans cette catégorie.

Réseaux Super-pair. Comme évoqué dans les sections précédentes, le réseau Super-pair profite de l'hétérogénéité des pairs en assignant aux pairs les plus puissants des rôles d'indexation, et par conséquent, de routage alors que les moins fortunés sont limités au rôle de fournisseurs de données. Le routage est fait en deux phases, d'abord entre les super-pairs qui forment un réseau avec une topologie pouvant être quelconque et ensuite entre un super-pair et les pairs qu'il administre. L'approche présentée dans cette thèse utilise la topologie Super-pair. D'autres PDMS pouvant être classés dans cette catégorie sont, par exemple, SQPeer[59], XPeer[103], TOPICS[69] et Edutella[82].

2.6.4 Médiation de données sémantique dans les PDMS

2.6.4.1 Généralités

Le problème de la médiation a déjà été évoqué dans les systèmes d'intégration de données. Cependant, étant donné les caractéristiques des systèmes P2P, la définition d'un schéma global médiateur unique est pratiquement impraticable pour les raisons suivantes :

(i) *Volatilité* : étant donné que les pairs peuvent joindre ou quitter le système à tout moment, le schéma global médiateur devrait être modifié en permanence pour refléter l'(in)disponibilité des données.

(ii) *Autonomie des pairs* : certains pairs peuvent publier toutes leurs données, tandis que d'autres seront intéressés par le partage d'une partie de leurs données.

(iii) *Passage à l'échelle* : La supposition d'un schéma global pose des défis supplémentaires quant à sa conception et à sa maintenance. La centralisation, en plus de nécessiter des ressources matérielles et réseaux supplémentaires, constitue un goulot d'étranglement. Les pairs ne peuvent pas changer de façon considérable, sans violer les correspondances antérieures établies avec le schéma médiateur. D'autre part, toute mise à jour sur le schéma central nécessite une coordination entre les pairs impliqués, ce qui augmente la complexité du système.

Deux techniques sont utilisées dans les systèmes P2P pour définir les correspondances sémantiques : *correspondances au niveau schémas* et *correspondances au niveau données*. Les correspondances au niveau schéma sont utilisées quand des schémas différents utilisent des noms et formalismes différents pour représenter les mêmes données. Les correspondances au niveau données sont utilisées quand les différences sémantiques entre schémas rendent l'établissement de correspondances au niveau schéma inapplicable. Par conséquent, ces deux approches sont complémentaires.

2.6.4.2 Correspondances entre schémas

Les correspondances sémantiques définissent des équivalences entre éléments d'un ou plusieurs schémas. Ces correspondances peuvent être ou ne pas être *un-à-un* ou réflexives. Néanmoins, elles sont transitives. Comme dans les bases de données distribuées, l'objectif visé avec l'établissement des correspondances sémantiques dans un système P2P, est la mise en place d'un environnement d'interrogation qui masque l'hétérogénéité et la distribution des sources de données.

A cause des contraintes citées ci-dessus, la plupart des PDMS évitent la maintenance d'un schéma unifié. A l'opposé, leurs approches se passent de la structure centrale et peuvent être classées en trois catégories : *médiation deux-à deux*, *médiation entre petits groupes*, *médiation avec super-pairs*. Les différentes catégories peuvent être décrites comme suit :

(1) *médiation deux-à-deux* : L'approche la plus simple pour implémenter les correspondances sémantiques dans un système P2P consiste à les définir uniquement entre couples de pairs. Ces correspondances sont stockées de part et d'autre de ces pairs intéressés par le partage de données. Dans certains cas, les correspondances sont établies et maintenues de façon manuelle (non-automatique) par les experts du domaine.

L'approche LRM (*Local Relational Model*)[15] suit ce procédé d'intégration de schémas. Dans cette approche, chaque pair spécifie des règles de transformation et des formules de coordination qui définissent comment son schéma est lié au schéma d'un autre pair, appelé *accointance*. Ces formules sont modifiées manuellement au cours du temps, à mesure que les exigences de partage de données évoluent. Les liens d'accointances entre pairs forment un réseau sémantique. Le système APPA[6] suppose que les pairs désireux de partager leurs données sont en accord sur une

description de schéma commune (CSD). Les schémas des pairs sont spécifiés comme des vues sur le CSD et les requêtes exprimées en termes des schémas locaux, et non du CSD. APPA suppose que ces correspondances sont maintenues jusqu'à ce que le partage de données ne soit plus désiré. Le *dialogue sémantique* proposé dans [1] suppose l'existence préalable de correspondances entre pairs et que ces correspondances sont conçues par des experts. En s'appuyant sur la transitivité de ces correspondances, les requêtes sont propagées vers les nœuds avec lesquels il n'y a pas de correspondance sémantique directe. Cette propagation permet de mesurer en même temps l'affinité sémantique avec les pairs pour lesquels il n'y a pas encore de processus de découverte de correspondances. En analysant les résultats des requêtes, les correspondances sont ajustées ou de nouvelles correspondances sont établies. L'objectif visé est d'établir de façon incrémentale une médiation globale entre les pairs.

(2) *médiation entre petits groupes* : Dans cette approche un pair peut définir des correspondances sémantiques impliquant deux ou plusieurs pairs et donc, elle constitue une généralisation des correspondances deux-à-deux.
Piazza[41] et PeerDB[84](cf. section 2.6.6.1) suivent cette approche d'intégration. Le système Piazza[41] assume que les pairs désireux de partager des données définissent et maintiennent des correspondances entre leurs schémas. PeerDB permet la création des correspondances sémantiques au moment de l'exécution de la requête. Les relations et les attributs sont décrits avec des mots-clés. Les requêtes sont diffusées à tous les voisins et les techniques de Recherche d'Informations, appliquées aux termes décrivant les relations et les attributs, permettent de déterminer si les relations et attributs locaux correspondent à ceux mentionnés dans la requête. L'utilisateur ayant initié la requête peut ainsi décider de l'exécution ou non de la requête sur le pair distant. Cette décision de l'utilisateur est mémorisée par le système et est utilisée plus tard pour toutes les requêtes faisant référence aux mêmes attributs.

(3) *médiation avec super-pairs* : Dans les systèmes super-pairs, les schémas réconciliés sont définis au niveau super-pair. Un tel schéma contient les correspondances sémantiques pour tous les pairs rattachés au super-pair. Il s'y ajoute la définition de correspondances sémantiques entre schémas de super-pairs, permettant ainsi d'implémenter le partage de données entre les pairs associés aux différents super-pairs.
Edutella[82] est un système qui implémente les correspondances au niveau super-pair.

Indépendamment des approches utilisées pour découvrir les correspondances sémantiques, les systèmes P2P tentent d'utiliser les relations transitives entre pairs pour réaliser le partage et l'intégration de données[72]. Contrairement aux systèmes distribués traditionnels dans lesquels les correspondances sémantiques induisent une topologie sémantique sous forme arborescente, dans le cas des PDMS, les correspondances sémantiques forment un *graphe sémantique* .

2.6.4.3 Correspondances entre données

La découverte de correspondances sémantiques entre schémas est adaptée au cas où les schémas à intégrer sont très liés sémantiquement. En d'autres termes, l'applicabilité concerne les cas où les différences entre les schémas sont principalement structurelles : les éléments de schémas représentent la même information ou peuvent être transformés en des éléments structurellement similaires dans d'autres schémas. Quand les éléments des schémas différent, mais sont tout de même sémantiquement liés, la définition des correspondances entre données est souvent le seul moyen de permettre le partage de données.

Les correspondances entre données sont implémentées comme des relations entre les attributs mis en correspondance. Ces relations de correspondance sont appelées *tables de correspondance* et représentent

la connaissance des experts[7]. Les tuples stockés dans la table de correspondance spécifient les correspondances entre les valeurs des relations pour lesquelles l'appariement est en train d'être effectué. Les tuples sont spécifiés par les spécialistes du domaine.

Les correspondances entre données, aussi bien que celles entre schémas, sont transitives, et parfois *un-à-un*. Elles définissent par ailleurs un graphe sémantique entre pairs.

2.6.5 Traitement des requêtes

Les techniques de traitement de requêtes dans les PDMS sont largement héritées des bases de données distribuées. Le processus du traitement de requêtes est constitué de trois phases principales : la *génération des plans de requête*, l'*optimisation* de ces plans et enfin leur *exécution*.

Le générateur de plans est chargé d'établir les plans de requêtes qui guident l'exécution distribuée de la requête en tenant compte des informations, sur les pairs pertinents, retournées lors de la phase de routage. En outre, le plan spécifie l'ordre d'exécution des opérateurs de la requête chez les pairs distants tout en privilégiant leur exécution au sein d'un même pair. La génération des requêtes peut être prise en charge par un seul pair ou un certain nombre de pairs qui vont coopérer. Dans le premier cas, la génération est basée sur une connaissance globale pour créer des plans optimaux et complets. Dans le second cas, cette connaissance est distribuée entre les pairs et seuls des plans de requêtes partiels peuvent être crées par les pairs. Contrairement aux DDMS dans lesquels les plans de requêtes sont statiques, l'adaptabilité des plans de requêtes aux départs des pairs durant l'exécution est une caractéristique principale des PDMS. L'adaptabilité peut être réalisée en changeant entièrement ou partiellement le plan d'exécution courant.

Le but de l'optimisation est de créer un plan d'exécution efficace en accord avec le temps d'exécution estimé de la requête. La plupart des méthodes d'optimisation sont basées sur des heuristiques ou sur un modèle de coût qui favorisent le plus possible l'évaluation au sein d'un même pair. Le meilleur plan est défini par rapport à un modèle de coût qui estime le total des ressources consommées par la requête ou son temps de réponse. La différence entre ces deux métriques de coût est que le second prend en compte la parallélisation dans l'exécution des plans. Ce coût tient compte du coût de traitement de chaque opérateur mais aussi le coût de transfert des données entre pairs pour exécuter la requête. La détermination du coût est faite à partir de statistiques, centralisées ou distribuées, sur les connections réseaux entre les pairs mais aussi sur leurs sources de données. La détermination du plan minimisant le coût est accompagnée d'un algorithme d'énumération qui liste tous les plans alternatifs possibles.

La phase d'exécution consiste à distribuer les sous-requêtes aux pairs appropriés et ensuite à renvoyer les résultats au pair initiateur de la requête. Cette exécution peut être centralisée ou totalement distribuée. Le premier cas correspond à l'approche traditionnelle dans les DDMS. Dans le dernier cas, plusieurs pairs communiquent entre eux pour exécuter le plan. Les résultats des sous-requêtes sont échangés entre pairs et quand la réponse est complète, elle est retournée au pair ayant initié la requête.

Plusieurs projets s'intéressent au problème de la génération des plans de requêtes dans un PDMS et à leur optimisation. Nous en citons quelques uns avant d'en faire une revue complète par système dans le paragraphe 2.6.6.1.

Dans [88], les auteurs introduisent les *plans de requêtes mutant* (MQP) qui sont des sérialisations XML de graphe de plans encapsulant des plans partiellement évalués et des données. Les plans sont mutés par les pairs en les ré-optimisant, les évaluant et en remplaçant les parties évaluées par les données correspondantes. Quand le plan est entièrement évalué, il est renvoyé au pair initiateur. Cependant, cette approche nécessite la diffusion dans tout le réseau du MQP, consommant ainsi beaucoup de bande passante en faisant migrer des fragments potentiellement volumineux de documents ou des fragments de

plans XML partiellement exécutés.

Dans [59], les auteurs décrivent un système P2P permettant d'évaluer des requêtes au format RQL (*RDF Query Language*) en utilisant la connaissance des schémas RDF. Ils se focalisent sur la construction et l'optimisation de plans de requêtes. L'algorithme de traitement de requêtes prend en entrée une requête annotée avec les pairs pouvant la traiter et produit un plan de requête en accord à la distribution des données.

Le projet Edutella [82] a pour but de concevoir une infrastructure P2P pour le Web Sémantique. Le processus de traitement de requêtes autorise des plans de requêtes contenant des prédicats de sélection, des fonctions d'aggrégations, des jointures, etc., et qui sont poussés vers les clients ou les super-pairs pour être exécutés. Les super-pairs disposent de modules d'optimisation pour générer les plans et déterminer les sous-plans qui vont être envoyés vers les pairs distants et ceux qui vont être exécutés localement.

Le système QueryFlow[58] permet le traitement dynamique et distribué de requêtes en utilisant la notion de *HyperQueries*. Les *HyperQueries* sont essentiellement des plans de requêtes détenus par les pairs et destinés à guider le routage et le traitement de requêtes dans le réseau.

ubQL[100] fournit un ensemble de primitives de manipulation de processus qui peuvent être ajoutés au dessus de n'importe quel langage de requête déclaratif. ubQL sépare la phase de déploiement et d'exécution d'une requête et permet l'adaptabilité des plans de requêtes durant la phase d'exécution.

2.6.6 Revue des systèmes existants

Dans les paragraphes suivants nous allons décrire des systèmes dont les travaux sont liés aux nôtres en insistant sur leurs aspects respectifs. La liste de ces systèmes n'est pas exhaustive et pourrait encore être étendue à d'autres systèmes. Le tableau 2.3 fournit une vue comparative des systèmes en termes de topologie, de modèle de données et de langage de requêtes.

Dans le tableau 2.4, nous proposons une classification des PDMS que nous allons présenter en nous basant sur leurs capacités en termes de routage et de génération de plans. Pour la génération des plans, nous distinguons les systèmes qui produisent des plans de façon centralisée et ceux qui les génèrent de façon distribuée. La différence vient du fait qu'un ou plusieurs pairs peuvent contribuer au processus de génération de plan.

Quant au routage des requêtes, nous considérons le cas distribué uniquement. Cela signifie que chaque pair maintient de façon indépendante un sous-ensemble des informations sur la localisation des autres pairs et, par conséquent, participe à la phase de routage. Les phases de routage et de génération de plans peuvent être séquentielles ou imbriquées. Le premier groupe de systèmes décrit dans le tableau 2.4 contient ceux dont les deux phases sont distinctes. Dans la deuxième famille, le routage et la génération des plans sont indissociables étant donné que le plan de requête est produit en plusieurs étapes en accord avec les informations sur les contenus des pairs qui peuvent être disponibles à un instant ou à un autre. Tous les systèmes dans les deux catégories utilisent le plus souvent des connaissances intentionnelles sur les schémas. Un autre critère de subdivision pourrait être la topologie, mais nous considérons que ce facteur est inclus dans les deux premiers considérés dans notre schéma de classification.

2.6.6.1 Systèmes orientés traitement de requêtes

PeerDB[84] permet le partage de données relationnelles distribuées sans partage de schéma. Il combine les propriétés des systèmes multi-agents avec celles des systèmes Pair-à-Pair. Chaque pair fournit une base de données relationnelle décrite grâce à des méta-données (mots-clés). Pour trouver les pairs pertinents à sa requête, un pair diffuse celle-ci à tous ses voisins. Chaque nœud recevant la

Système	Topologie	Modèle de données	Langage de requête	Ref.
Piazza	non-structuré	XML	Sous-ens. XQuery	[41]
PeerDB	non-structuré	Relationnel	SQL	[84]
SQPeer	Super-pair ou DHT	RDF	RQL	[59]
XPeer	Hybride	XML	Sous-ens. XQuery	[103]
Edutella	Super-pair	RDF	RDF-QEL	[82]
PIER	DHT (Bamboo)	Relationnel	-	[47]
Bibster	Raccourcis	RDF	SeRQL	[23]
Humboldt Discoverer	Hybride	RDF	SPARQL	[45]
RDFPeers	DHT (MAAN/Chord)	RDF	RDQL	[24]
ubQL	non-structuré	-	-	[100]
APPA	structuré/non-structuré	XML	XQuery	[6]
MQP	non-structuré	XML	-	[88]
AmbientDB	DHT(Chord)	Relationnel	SQL	[17]
iXPeer	non-structuré	XML/RDF/Relationnel	-	[11]
QueryFlow	-	Relationnel	SQL	[58]
PEPSINT	Hybride	XML/RDF	c-XQuery/c-RDQL	[28]
SEWASIE	non-structurée	XML/Relationnel	-	[14]
SomeWhere	non-structuré	OWL	-	[5]
Semantic Agreement	non-structuré	OWL	SPARQL	[114]
Hyperion	non-structuré	Relationnel	SQL	[7]
GridVine	DHT (P-Grid)	RDF/S	RDQL	[2]
FREddies	DHT(PIER)	Relationnel	SQL	[48]
SwAP	-	Relationnel	SQL	[118]

Table 2.3 – Les PDMS vus sous différents aspects.

requête fait une mise en correspondant des mots-clés décrivant les relations de la requête avec ceux décrivant les relations qu'il détient. Les relations pertinentes vont alors être renvoyées au pair initiateur qui va réécrire la requête et les envoyer au pairs pertinents. La reformulation des requêtes est faite par des agents. En se basant sur les statistiques, les connections avec les voisins sont adaptées au fil du temps. Il n'y a pas de processus classique de découverte de correspondances entre les schémas des pairs. L'approche PeerDB présente la faiblesse d'autoriser des correspondances entre mots-clés pouvant aboutir à de fausses reformulations de requêtes. L'utilisateur doit décider quelles requêtes ont un sens et quelles requêtes doivent être exécutées. Notons aussi que la version actuelle de PeerDB ne supporte que les données relationnelles et qu'elle a été évaluée sur un réseau de 32 nœuds.

XPeer [103] est basé sur une architecture hybride permettant le partage de données XML concernant n'importe quel domaine. L'intégration se fait sans schéma global, ce qui réduit considérablement les tâches d'administration. Chaque pair exporte la description des données à partager sous forme d'une arborescence qui est automatiquement inférée des données. Les pairs sont organisés logiquement en groupes sur la base de critères de similarité de schémas. Le schéma d'un super-pair est construit sans activité d'intégration, ce qui fait que l'assistance humaine n'est pas nécessaire. Les requêtes sont écrites dans un sous-ensemble de XQuery. Cependant l'applicabilité de XPeer concerne seulement les situations pour lesquelles la découverte des correspondances peut être évi-

	Générateur de Plan Centralisé	**Générateur de Plan Distribué**
Routage de Requêtes Distribuées	*PDMSs avec Regroupement Sémantique* Edutella[82], Humboldt Discoverer[45] XPeer[103]	*PDMSs Orientés Mappings* Piazza[41], Hyperion[7], iXPeer[11] GridVine [2], PEPSINT[28]
	PDMSs Structurés AmbientDB[17][21]RDFPeers[24], SQPeer[59]	*PDMSs avec Plans Mobiles* QueryFlow[58], MQP[88], ubQL[100]
		PDMSs avec Plans Adaptatifs FREddies[48], SwAP[118]

Table 2.4 – Classification des systèmes du point de vue routage de requêtes et génération de plans

tée ou être faite en dehors du système Pair-à-Pair.

Edutella[82] fournit une infrastructure P2P supportant les méta-données RDF. Il permet l'échange de méta-données du domaine éducatif en utilisant les schémas IEEE LOM, IMS et ADL SCORM pour décrire les méta-données d'apprentissage en ligne. Les données sont décrites relativement à des ontologies de référence (e.g., http://dmoz.org) et elles peuvent être stockées dans une base de données ou un document XML. Le réseau Edutella est basé sur une topologie Super-pair et dans laquelle les super-pairs sont organisés en hyper-cube pour router les requêtes. Les données sont échangées dans un format commun (*Edutella Common Data Model*) et les requêtes dans un format commun d'échange de requêtes appelé RDF-QEL. Edutella ne fait pas de découverte classique de correspondances sémantiques car ne définissant que des correspondances syntaxiques (médiation syntaxique) au niveau des langages de requêtes. D'autre part, Edutella procède par inondation au niveau super-pair. Le processus de traitement de requêtes autorise des plans de requêtes contenant des prédicats de sélection, des fonctions d'aggrégations, des jointures, etc. Ces plans sont poussés vers les clients ou les super-pairs pour être exécutés. Chaque super-pair dispose d'un module d'optimisation pour générer les plans et déterminer les sous-plans qui vont être envoyés vers les pairs distants et ceux qui vont être exécutés localement.

SQPeer[59] Le système SQPeer utilise un réseau sémantique sous-jacent (*Semantic Overlay Network*-SON), formé en regroupant les pairs partageant des informations similaires sur leurs schémas. Dans SQPeer chaque pair détient une source de données au format RDF en accord avec des schémas RDF. Les requêtes sont exprimées en RQL (*RDF Query Language*), un langage du style SQL pour RDF. Chaque pair publie une vue RVL (*RDF View Language*) décrivant son schéma. Ces vues sont diffusées dans tout le réseau P2P. Une topologie sémantique permet de regrouper les pairs partageant des schémas similaires. Chaque requête est comparée avec les vues locales détenues par le pair et ensuite elle est annotée avec les informations sur la localisation des pairs pertinents. Le plan de requête est ensuite généré puis distribué et les résultats fusionnés par la suite. SQPeer supporte aussi bien une topologie super-pair que DHT pour l'organisation des vues (schémas). Cette approche nécessite une imbrication des phases de routage et de génération de plans étant donné que les informations nécessaires pour la génération du plan ne sont pas entièrement disponibles sur le nœud et elles doivent donc être collectées.

PIER[47][46] a pour objectif de fournir un processeur de requête relationnelle distribué en se basant sur les techniques P2P. A cet effet, les DHTs (CAN[93] dans un premier temps et maintenant Bambo[94], un dérivé de Pastry[98]) sont utilisées pour indexer les tuples stockés dans le système. Chaque tuple est transmis au nœud responsable de l'intervalle de sa clé. Ainsi donc, les données ne restent pas dans les nœuds fournisseurs, mais elles sont distribuées en accord avec l'algorithme

DHT. Pour chaque attribut indexé, un espace de nommage DHT séparé est utilisé. PIER supporte aussi bien les jointures, les sélections que les aggrégations sur les tuples. Actuellement, les tuples sont stockés en mémoire, ce qui limite la capacité de stockage du réseau.

Bibster[23] permet l'échange de données bibliographiques entre chercheurs. Il est basé sur deux ontologies permettant de structurer automatiquement les données des pairs, se passant ainsi du problème de découverte des correspondances. Les ontologies interviennent dans le stockage des données, la reformulation et le routage des requêtes mais aussi dans la présentation des résultats. La sélection des pairs pertinents est basée sur des descriptions d'expertise des pairs, induisant la formation d'un réseau Pair-à-Pair sémantique indépendant de la topologie Pair-à-Pair existante. Les requêtes sont formulées en termes des deux ontologies et routées de façon intelligente en fonction des descriptions d'expertises connues par le pair actuel. Bibster ne supporte que les entrées BibTeX. De plus, Bibster ne constitue pas une architecture flexible car fournissant une application spécifique ne fonctionnant que pour le domaine bibliographique. De ce fait, avec le changement des métadonnées, la réutilisation est compromise.

Humboldt Discoverer[45] réalise une architecture P2P hybride combinant une approche DHT avec un réseau P2P non-structuré et dans laquelle les données stockées par les pairs se conforment uniquement au modèle relationnel. Pour éviter le processus classique de découverte de correspondances sémantiques dans les PDMS les pairs décrivent leurs schémas en accord avec des ontologies connues. Chaque schéma est alors exporté sous forme d'un graphe RDF contenant les concepts et les propriétés couvertes par le pair concerné. Les correspondances sémantiques entre les schémas des pairs et les ontologies sont pré-établies (en utilisant l'approche GLAV), et elles sont faites manuellement. La détection des pairs pertinents est guidée par des pairs spéciaux, appelés consultants et experts, et elle repose sur un index P2P sémantique renfermant les informations sur les pairs distants. Cette recherche repose sur une comparaison entre graphe de requête et graphe de pair et est ramenée à un problème d'appariement de graphes. Le processus de traitement de requête n'a pas été entièrement défini.

RDFPeers[24] a pour but de construire un entrepôt RDF distribué à grande échelle et dans lequel chaque nœud peut stocker ou rechercher des triplets RDF de manière transparente. RDQL est utilisé pour interroger les bases des pairs et pour ce faire, il opère au niveau des triplets RDF sans prendre en compte les informations sur les schémas. Les pairs sont organisés en cercle virtuel, de façon similaire à l'anneau Chord. Quand un triplet RDF est inséré dans le réseau, il sera stocké dans trois pairs par application d'une fonction de hachage sur le sujet, l'attribut et la valeur de l'objet. Ainsi, des requêtes de triplets peuvent être routées efficacement vers les pairs censés les contenir au cas ou ils existeraient. Une requête est supposée contenir plusieurs sous-requêtes, une pour chaque attribut impliqué. La sous-requête la plus sélective est utilisée pour un routage initial de la requête dans la DHT, ensuite les tuples pertinents sont renvoyés au nœud responsable du prochain critère de sélection, ainsi de suite, jusqu'à ce que l'intersection entre tous les critères de sélection des sous-requêtes soit examinée.

ubQL[100](*ubiquitous Query Language*), fournit un ensemble de primitives qui peuvent être ajoutées au dessus de n'importe quel langage de requête traditionnel dans le but de permettre l'optimisation de requêtes distribuées. Les requêtes sont encapsulées dans des processus qui peuvent migrer et être configurés entre pairs. ubQL sépare le déploiement et la migration des processus d'une requête, de son exécution. Chaque pair contient des vues intentionnelles exprimées comme des processus répliqués, mais aussi des données à partager avec le reste du système. Des canaux de communication permettent la communication et l'échange de données entre pairs. La phase de déploiement est res-

ponsable du routage du processus de la requête et de l'assemblage des informations nécessaires à son exécution. A la réception de la requête, un pair a le pouvoir de la ré-optimiser avant d'envoyer les sous-requêtes correspondantes pour un traitement ultérieur par les pairs distants. L'optimisation considère l'ordonnancement simple des jointures si seulement des jointures locales sont concernées. Le modèle de coût utilisé considère à la fois le coût de traitement et de transmission. Enfin, ubQL supporte l'adaptabilité des plans de requête durant la phase d'exécution.

MQP Dans [88], les auteurs introduisent le concept de plan de requête mutant (*Mutant Query Plan*-MQP). Un plan de requête mutant est une sérialisation XML d'un graphe représentant un plan de requête et encapsulant des plans de requêtes partiellement évalués et des données. Les MQPs sont des plans de requêtes logiques dans lesquels les feuilles peuvent être des références URN/URL ou des données XML matérialisées. Les références vers les localisations des ressources(URLs) pointent vers les pairs dans lesquels les données actuelles résident, tandis que les noms abstraits de ressources (URNs) peuvent être vus comme les thèmes des données recherchées. De plus, chaque pair peut évaluer et ré-optimiser un fragment de MQP en ajoutant des fragments de XML dans les feuilles et ensuite le router vers un autre pair. Quand un MQP est entièrement évalué, c'est-à-dire ramené en un document XML concret, le résultat est renvoyé au pair source qui a initié la requête. Dans le cas contraire, il est transmis au pair suivant pour continuer le processus de traitement. Le routage reste efficace et s'appuie sur des catalogues distribués contenant les informations sur la localisation des thèmes recherchés. Cependant, cette approche nécessite la diffusion dans tout le réseau du MQP, consommant ainsi beaucoup de bande passante en faisant migrer des fragments potentiellement volumineux de documents ou des fragments de plans XML partiellement exécutés.

AmbientDB[17] traite le problème de la gestion de données relationnelles dans un contexte P2P. Il assume l'existence d'un schéma global avec la possibilité pour chaque pair d'avoir son propre schéma pourvu qu'il fournisse les correspondances avec le schéma global. Un protocole P2P initialisé dans chaque pair est responsable du routage des requêtes et utilise Chord[106] pour connecter les pairs en implémentant une table d'index distribuée entre les pairs. Chaque pair contient un processeur de requêtes qui est responsable de l'exécution des requêtes sur des données locales ou distribuées. Une requête formulée par un pair est exprimée dans l'algèbre relationnel standard, puis le choix de l'ordre d'exécution des opérateurs et de leurs lieux d'exécution est opéré. Différentes stratégies d'exécution des jointures sont aussi présentées. Une approche Super-pair pour AmbientDB est présentée dans [21].

QueryFlow[58] permet un traitement dynamique et distribué des requêtes avec le concept de *HyperQueries*. Les *HyperQueries* sont essentiellement des sous-plans qui existent dans les pairs et qui sont destinés à guider le processus de traitement de requêtes à travers le réseau. L'approche proposée est basée sur des attributs virtuels exprimés en XML et dont les valeurs sont déterminées en évaluant les requêtes distantes. La notion de *Hyperlinks*, qui sont des URIs, est utilisée pour le routage des requêtes. Les requêtes sont d'abord exprimées sous forme de plans avec les opérateurs. Les sous-plans locaux sont exécutés par le pair initiateur et ceux qui restent sont diffusés dans le réseau et leur exécution ainsi déléguée à ces pairs distants. Des techniques d'optimisation telles que la collecte des résultats par des nœuds intermédiaires distants sont utilisées. Les expérimentations montrent le passage à l'échelle du système en accord avec le nombre de clients et d'objets participant au système.

FREddies[48] opère dans le framework PIER[47], un processeur de requête basé sur les DHTs. Un FREddy est un opérateur qui route dynamiquement les tuples vers des opérateurs locaux, lesquels peuvent les envoyer vers d'autres pairs à travers le réseau. Une DHT, utilisée par PIER, organise

les pairs en réseau sous-jacent pour le routage, tout en indexant directement les données. Chaque requête est associée à un plan qui détermine les opérateurs qui sont nécessaires pour l'exécuter, mais aussi l'ordre dans lequel ils devront être routés. Tous les nœuds participant à l'exécution sont au courant du même plan. Quand un opérateur arrive à un nœud donné, tous les opérateurs, y compris le FREddy sont instanciés. Ainsi donc, le FREddy débute la transmission du flux de données en demandant les tuples de chacune de ses sources. Les méta-données de chaque tuple sont utilisées pour identifier les opérateurs à travers lesquels le tuple est déjà passé. Des stratégies de routage basées sur un ordre spécifique des opérateurs ou sur le hasard en accord avec les entrées de chaque opérateur sont proposées. Ces stratégies déterminent l'ordre du traitement des requêtes jusqu'à l'exécution complète de la requête. L'optimisation à l'exécution est aussi évoquée.

SwAP[118] est un système distribué dans lequel l'adaptabilité des plans durant la phase d'exécution est intégrée. Chaque pair contribuant au plan de la requête est équipé d'un mécanisme fournissant l'adaptabilité du plan durant la phase d'exécution. Une phase préparatoire réalisée par un pair coordonnateur produit un graphe de traitement distribué. Ce graphe contient tous les pairs participant au traitement des opérations, mais aussi les informations sur les types de parallélisme à utiliser. Ensuite, le traitement du plan est lancé et les opérateurs appropriés pour la communication entre pairs initialisés. Les tuples sont routés vers chaque pair pour être traités jusqu'à ce que ce dernier termine l'exécution des opérations nécessaires. Au sein de chaque pair, un routage ultérieur de chaque requête est effectué en accord avec les statistiques obtenues à l'exécution. Ces statistiques sont assemblées en échangeant des tuples virtuels entre pairs. Ces tuples virtuels ne contiennent pas de données mais fournissent un moyen de superviser l'exécution de la requête.

2.6.6.2 Systèmes orientés gestion de méta-données

Dans les PDMS l'utilisation des ontologies et des techniques du Web sémantique a été identifiée comme très prometteuse. Dans ces systèmes, le contenu des nœuds est décrit relativement à une ontologie/taxonomie qui permet de localiser les nœuds sémantiquement pertinents pour une requête. Dans la suite nous présentons quelques uns de ces PDMS.

Piazza[41] est basé sur une architecture non-structurée permettant l'échange de données XML ou relationnelles. En présence de différents schémas et de différentes représentations, les pairs intéressés par l'échange de données définissent des correspondances sémantiques entre eux, deux à deux ou entre petits groupes de pairs. Chaque pair exprime ses requêtes sur son propre schéma. Les requêtes sont dans ce cas évaluées globalement sur un réseau de pairs sémantiquement liés par les correspondances. Piazza combine et généralise les formalismes LAV (*Local-As-View*) et GAV (*Global-As-View*) proposées dans la médiation de schémas dans les systèmes d'intégration de données et les étend aux documents XML. Le langage d'expression des correspondances pour les données relationnelles est PPL (*Peer Programming Language*) tandis que celui utilisé pour les documents XML est basé sur XQuery. La réécriture des requêtes est basée sur une comparaison entre les expressions XQuery et les correspondances sémantiques et elle est faite de manière centralisée. L'approche Piazza peut aussi être appliquée aux données RDF avec cependant une expressivité limitée. L'approche Piazza présente des insuffisances liées à la difficulté de décrire les correspondances, de les construire mais aussi à la maintenance de ces dernières. D'autre part, il n'a pas été donnée une évaluation des performances de Piazza.

Hyperion[7] propose une architecture pour un PDMS dans lequel la publication de chaque pair est une base de données relationnelle et dans laquelle il n'y a pas de schéma global. L'échange d'informations entre pairs est possible grâce à la définition de tables de correspondance et d'expressions

de correspondances qui stockent les correspondances sémantiques entre éléments des schémas des pairs. Un gestionnaire de requêtes utilise les tables et les expressions de correspondances pour réécrire une requête exprimée sur le schéma d'un pair spécifique sur les schémas des pairs liés. Cependant, les tables de correspondance, bien que plus faciles à établir qu'un processus complet de découverte de correspondances entre schémas, sont jusqu'ici créées manuellement par des spécialistes du domaine ce qui peut être coûteux en temps.

iXPeer[11] a été proposé à la suite de XPeer[12] (différent de XPeer[103]) et qui a pour but l'intégration de données dans un contexte P2P. Dans XPeer le réseau est organisé logiquement sur la base de la sémantique des données, ce qui va guider le traitement des requêtes et l'échange de données entre pairs. Les pairs sont classés en catégories suivant les fonctions offertes au réseau, avec comme objectif de réduire le nombre de correspondances à spécifier entre sources de données de pairs, ce qui augmente le passage à l'échelle du système. iXPeer est une extension de XPeer venant à bout des fautes dans XPeer. Les auteurs utilisent les primitives P2P offertes par Automed[20]. Ce dernier fournit des outils permettant d'implémenter un modèle d'intégration de données dans un contexte P2P. L'outillage ne fournit pas une approche fixe sur comment procéder à l'intégration mais plutôt des méthodes générales qui peuvent être utilisées pour implémenter un certain nombre d'architectures P2P. Étant donné que iXPeer supporte n'importe quel modèle de données structurée ou semi-structurée, son utilisation rend iXPeer plus flexible. L'implémentation actuelle de Automed supporte déjà les données relationnelles, XML, Entité/Relationnel, RDF et les fichiers textes semi-structurés tels que CVS.

PEPSINT[28] permet d'intégrer sémantiquement des sources de données XML et RDF hétérogènes dans un environnement Pair-à-Pair. Le système est basé sur une architecture Pair-à-Pair hybride composée d'un super-pair unique et d'un ensemble de pairs. Le super-pair en question contient une ontologie RDF globale (construite en utilisant l'approche GAV). Les pairs abritent les schémas locaux et les sources de données locales. Cette ontologie sert non seulement de point de contrôle central sur l'ensemble des pairs du réseau mais aussi de médiateur pour la réécriture des requêtes d'un pair à un autre. Chaque pair connecté au réseau est indexé par le super-pair grâce à un certain nombre de correspondances sémantiques stockées dans une table de correspondance qui indique les équivalences entre les éléments des schémas. Les correspondances sont établies grâce à un processus classique d'appariement de schémas. Durant cette opération, l'ontologie globale est étendue par intégration des schémas locaux. Les requêtes sont reformulées par le super-pair en utilisant des compositions de correspondances du pair initial vers le super-pair et du super-pair vers les autres pairs. Un défi majeur à relever pour PEPSINT est la tolérance aux fautes. En effet, avec l'utilisation d'un super-pair central sollicité pour tout traitement de requêtes, le système ne fonctionne plus en cas d'indisponibilité de ce dernier.

SEWASIE[14] se propose de permettre aussi bien l'échange de données structurées, semi-structurées que non structurées. Chaque pair contient une information spécifique à propos des domaines concernés, mais seulement une couche de cette connaissance doit être exportée vers les autres pairs à travers un langage standard pour représenter la structure de ces sources. La connaissance en question est représentée à travers une ontologie. La connection entre les pairs repose alors sur l'échange de méta-données XML. L'intégration est possible grâce à la création d'une vue virtuelle globale des sources de données et d'un certain nombre de correspondances entre cette vue et les sources intégrées. Un gestionnaire de requêtes décompose la requête en tenant compte des correspondances entre la vue virtuelle globale et les sources pouvant répondre à la requête, envoie les requêtes, collecte les résultats et les envoie au pair interrogateur.

Semantic Agreement[114] Dans cette approche les auteurs étudient les problèmes liés à la gestion et à la médiation des données dans un contexte P2P. L'approche proposée permet une interopérabilité sémantique des sources d'informations en combinant les avantages de la médiation sémantique et ceux des systèmes P2P. Elle est basée sur une approche P2P pure avec une architecture Super-pair autorisant deux types de pairs. Le super-pair contient une ontologie de référence fournissant une ontologie commune (CO) du domaine. Les descriptions des ontologies avec leurs concepts, classes et relations sont faites avec OWL. Un pair dispose d'un schéma exporté (ES) représentant ses données locales. L'approche est basée sur un processus appelé *semantic agreement* et permettant l'établissement de correspondances sémantiques entre CO et ES. Ceci nécessite la définition et le calcul de mesures de similarité sémantique entre les concepts utilisés par les différents pairs. Ces correspondances sont exprimées comme des *unités d'agréments* et sont utilisées pour localiser les sources d'informations, mais aussi pour échanger de l'information entre les pairs.

SomeWhere[5] est un PDMS sémantique basé sur des ontologies distribuées à grande échelle. Le système repose sur une architecture pair-à-pair pure. Le modèle de données supporté est un fragment du langage d'ontologie Web OWL. Les pairs dont les données reposent sur d'autres modèles de données ne sont pas supportés, ce qui n'autorise pas d'autres langages de requêtes. Chaque pair joint le réseau à travers un ensemble de pairs connus appelés *accointances* et avec lesquels il déclare un certain nombre de correspondances. Les requêtes sont des combinaisons logiques des ontologies concernées et leur traitement est équivalent à raisonnement distribué sur des théories de logique propositionnelle.

GridVine[2] est l'un des rares PDMS orientés gestion de méta-données qui utilise une DHT comme topologie sous-jacente. Les données sont exprimées en RDF/S et RDQL est utilisé pour formuler les requêtes. L'indexation et la recherche des triplets RDF sont effectuées de la même façon que dans RDFPeers. Chaque triplet est indexé séparément sur son sujet, son prédicat et son objet. Les pairs sont autorisés à créer des correspondances sémantiques entre leurs schémas et ceux des pairs distants. Une méthode appelée dialogue sémantique (*semantic gossiping*) a été introduite pour permettre l'interopérabilité sémantique dans un cadre décentralisé. Les correspondances sémantiques sont utilisées pour réécrire de façon appropriée les requêtes entre les différents schémas. Une première évaluation expérimentale a été effectuée sur un réseau composé de 60 pairs.

APPA (ATLAS Peer-to-Peer Architecture)[6] est un système Pair-à-Pair de gestion de données distribuées ayant une architecture indépendante du type de réseau Pair-à-Pair (non structuré, DHT, Super-pair, etc.). Cette architecture repose sur des services organisés par niveaux (services de réseau Pair-à-Pair, services de base, services avancés). Les services avancés permettent le partage de données sémantiquement riches incluant la gestion des schémas, la réplication, le traitement des requêtes, etc. Les données partagées sont au format XML et le langage d'interrogation est XQuery. De plus, chaque pair a la possibilité de manipuler ses données XML localement à travers un adaptateur si besoin. Le partage de données sémantiquement riches se fait de façon décentralisée. APPA suppose que les pairs désireux de partager leurs données s'accordent sur une description de schéma commune (CSD). Le schéma d'un pair est exprimé comme une vue sur le CSD. Les requêtes sont exprimées en terme du schéma local et non du schéma commun. De plus, APPA considère que les correspondances entre schémas sont maintenues jusqu'à ce que le partage de données ne soit plus souhaité. Le système APPA est en cours d'implémentation avec le réseau générique JXTA.

2.7 Conclusion

Dans ce chapitre, nous avons présenté les principales technologies de partage de données et plus spécifiquement, les systèmes pair-à-pair de gestion de données distribuées qui sont un domaine de recherche très prometteur. Nous avons vu qu'un certain nombre de systèmes et d'approches, en relation avec ce thème, ont déjà été proposés dans la littérature.

Nous avons identifié deux tendances dans les PDMS. D'un côté, le schéma est supposé être homogène et les principaux défis à relever sont ceux du traitement des requêtes et de la génération des plans supervisant l'exécution des requêtes. De l'autre côté, les schémas sont supposés hétérogènes et, par conséquent, l'intégration des schémas devient le principal défi. Dans le dernier cas, des correspondances sémantiques sont utilisées pour passer d'un schéma à l'autre. Cependant, la capacité à passer à l'échelle devient restreinte. Quelques uns des systèmes se focalisant sur les schémas homogènes passent mieux à l'échelle. Par ailleurs, il est très difficile de comparer l'efficacité de ces systèmes premièrement parce qu'il n'existe pas de repère standard et deuxièmement parce que les modèles de données et l'expressivité des langages de requêtes varient largement d'un système à l'autre.

Nous n'avons pas l'intention, dans ce travail, de produire un substitut au travail établi dans les domaines de l'intégration de données et du traitement de requêtes distribuées. En effet, nous avons besoin d'utiliser certains de ces travaux à chaque fois qu'ils s'appliquent aux problèmes qui se posent à nous pendant le partage des ressources (médiation de données, optimisation de requêtes, etc.). Cependant, nous avons constaté que la plupart des PDMS n'insistent pas sur le cas où le réseau de partage de données autorise les pairs à publier des données conformes à un modèle de données quelconque et par conséquent l'utilisation de plusieurs langages de requêtes. De plus, il est supposé, dans certain cas, une homogénéité de schémas, étant donné que les données sont parfois décrites par rapport à des ontologies de référence.

Par conséquent, notre approche devient complémentaire à celles déjà proposées. Le but de notre travail est donc la médiation de données et le traitement des requêtes à l'échelle d'un système P2P de gestion de données distribuées supportant plusieurs modèles de données et autorisant plusieurs schémas/ontologies.

CHAPITRE 3

Médiation de données sémantique

3.1 Introduction

Dans ce chapitre, nous présentons l'architecture du réseau SenPeer, la structure fonctionnelle des nœuds du réseau ainsi que le processus de médiation de données sémantique. SenPeer suit une topologie super-pair reposant sur une organisation des pairs en communautés sémantiques en rapport avec leurs thèmes intérêts. La diversité des pairs en termes de modèles de données et de vocabulaires fait apparaître un problème d'hétérogénéité syntaxique et sémantique dans le réseau de partage de donnée. La médiation sémantique permet de faciliter le partage de données en réconciliant les différentes sources de données pouvant se conformer à des modèles de données quelconques. Cette réconciliation est basée sur la connaissance sémantique véhiculée par les schémas et elle passe par l'établissement de mesures de similarités sémantiques entre les éléments de schémas de pairs. Le processus de réconciliation sémantique, en combinaison avec les connaissances sur les schémas des pairs, induit une topologie sémantique au dessus de la topologie physique actuelle. Ce réseau sous-jacent sert de support à un mécanisme de routage intelligent des requêtes à destination des pairs pertinents, permettant ainsi de diminuer les efforts de traitement des requêtes.

3.2 Notre contexte

La conception d'un PDMS qui fournit des fonctionnalités similaires à celles supportées par une base de données distribuée semble encore surréaliste. Par conséquent, pour introduire une approche efficace de partage de données, nous avons besoin de faire certains compromis.

Nous nous plaçons dans le cadre des applications de partage de données pour lesquelles ces dernières peuvent être organisées par thèmes grâce à une taxonomie. Chaque pair est libre d'exporter des données de son choix à condition qu'elles soient décrites par un schéma. Les thèmes ne sont pas forcément disjoints et chacun correspond à une *communauté sémantique* particulière. Pour le cas du fleuve sénégal, les communautés identifiées comme intéressantes pour la mise en valeur de la vallée peuvent être représentées en partie dans la figure 3.1. Par exemple, les thèmes *hydrologie* et *maladies hydriques* sont non disjoints car les maladies hydriques sont liées au débit et au niveau de l'eau notamment dans les environs des barrages. De ce fait, les experts de ces deux communautés doivent disposer des données liées à l'eau.

La notion de communauté que nous introduisons ici permet de regrouper un ensemble d'utilisateurs autour d'un même nœud en accord avec leur communauté d'intérêt. Une communauté est définie, de façon abstraite, à travers un thème caractérisant le domaine d'intérêt. La définition des thèmes est établie hors du système et ces derniers sont dans un espace unique permettant de les comparer entre eux en vue

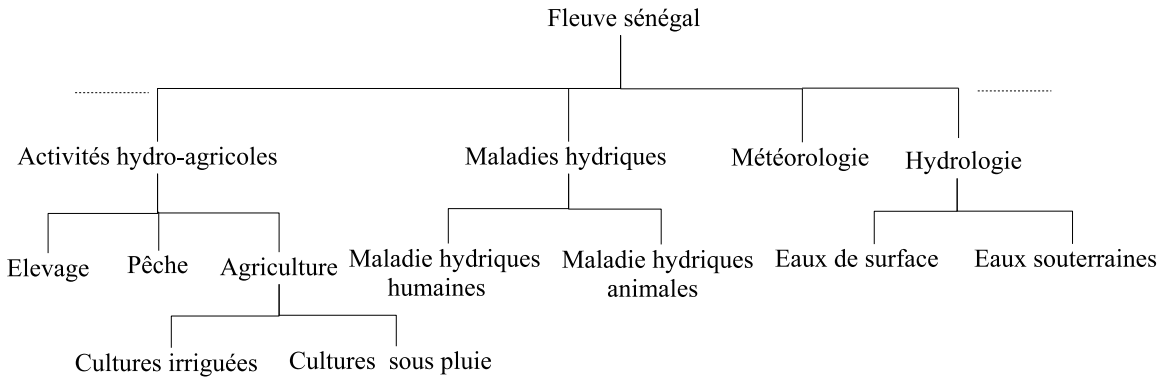


Figure 3.1 – Partie de la taxonomie organisant les données de la vallée du fleuve sénégal.

d'établir des liens. Nous supposons, de plus, que l'appartenance d'un utilisateur final à une communauté d'intérêt est clairement établie, ce qui permet d'éviter sa découverte par des techniques de regroupement sémantiques.

Nous considérons donc un PDMS dans lequel chaque pair détient une source de données. Nous supposons aussi un environnement non volatile qu'un pair peut, occasionnellement mais pas fréquemment, joindre ou quitter. Notre but principal est le partage transparent et efficace de données entre pairs.

3.3 Architecture de SenPeer

3.3.1 Topologie du réseau

Dans le but de permettre l'autonomie des pairs et l'auto-organisation du réseau, nous choisissons un réseau Super-pair non structuré. La topologie Super-pair combine l'efficacité de la recherche centralisée avec l'autonomie, l'équilibrage des charges et la robustesse de la recherche distribuée. L'architecture super-pair permet de profiter de l'hétérogénéité des pairs en assignant plus de responsabilités aux pairs capables de les assumer. Par conséquent, certains pairs, appelés super-pairs, qui ont un pouvoir de calcul additionnel et une plus grande bande passante effectuent des tâches administratives. Ils sont chargés de la gestion d'un ensemble de pairs, permettant entre autre de diminuer les efforts de compilation de requêtes mais aussi d'empêcher l'inondation du réseau par les requêtes. Le réseau de super-pairs constitue l'épine dorsale du réseau P2P prenant en charge le routage des messages et la médiation entre les méta-données. La topologie exacte du système dépend du processus de réconciliation, en respect avec le nombre super-pairs disponibles fournissant les garanties de bande passante et de connexion du système.

Dans chaque communauté se trouve un super-pair appelé *parrain* de la communauté. Les pairs sont connectés aux super-pairs en fonction de leurs thèmes d'intérêts ou communautés sémantiques (Figure 3.2). Chaque super-pair SP_j parrain d'une communauté sémantique C_j *suggère* un schéma SS_j pour cette communauté. Le schéma *suggéré* est purement conceptuel, les données nécessaires pour répondre aux requêtes étant stockées dans les pairs de la communauté. D'autre part, chaque pair est libre de décrire ses données avec son propre schéma S_i, pouvant être conforme à un modèle de données quelconque, et donc avec son propre vocabulaire. De plus, chaque pair P_i soumet des requêtes avec son propre langage d'interrogation.

En migrant l'architecture centralisée, les PDMS risquent de tomber dans une certaine anarchie si tous les pairs participant au système distribué se comportent comme ils veulent. Un point important dans notre travail est que les pairs doivent être en accord sur quelque chose. A l'instar des sociétés humaines, un langage commun est une première étape pour éviter l'anarchie. Pour ces raisons, nous utilisons : (i) un format d'échange de schémas, appelé $sGraph$, et qui permet d'échanger des schémas sémantiquement associés à travers le réseau sans faire aucune supposition sur les modèles de données des pairs ; (ii) un format d'échange de requêtes appelé SQUEL et qui permet l'échange de requêtes entre pairs. Toutes ces considérations seront prises en compte dans la réconciliation sémantique des pairs, le routage sémantique des requêtes dans le réseau ainsi que dans le traitement de ces mêmes requêtes.

Formellement, notre PDMS est un couple $\mathfrak{P} = (\mathcal{G}, \mathcal{M})$ avec

- $\mathcal{G} = \{(\mathcal{SP}, \mathcal{L}_{sp})\} \cup \{\mathcal{SP} \cup \mathcal{P}, \mathcal{L}_p\}$ est un graphe non orienté où
  - $\mathcal{SP} = \{SP_1, \ldots, SP_n\}$ est un ensemble de super-pairs, chacun avec son propre schéma,
  - $\mathcal{L}_{sp} = \{(SP_i, SP_j) | SP_i, SP_j \in \mathcal{SP}^2\}$ est un ensemble de liens sémantiques entre super-pairs (entre communautés),
  - $\mathcal{P} = \{P_1, \ldots, P_m\}$ est un ensemble de pairs, chacun avec son propre schéma et
  - $\mathcal{L}_p = \{(SP_i, P_j) | SP_i \in \mathcal{SP}, P_j \in \mathcal{P}\}$ est un ensemble de liens sémantiques entre pairs et super-pairs (lien d'appartenance à une communauté) ;
- $\mathcal{M} = \{\mathcal{M}^{ij}_{SP/SP}\} \cup \{\mathcal{M}^{kl}_{SP/P}\}$ est un ensemble de matrices de correspondances (entre vocabulaires différents), chacune associée à un lien sémantique $sl(i, j)$ ou $sl(k, l) \in \{\mathcal{L}_{sp}\} \cup \{\mathcal{L}_p\}$ et stockant les correspondances sémantiques entre schémas de (super-)pairs.

De plus, nous supposons que tous les pairs et super-pairs ont des identifiants uniques lesquels seront utilisés pour localiser les nœuds correspondants.

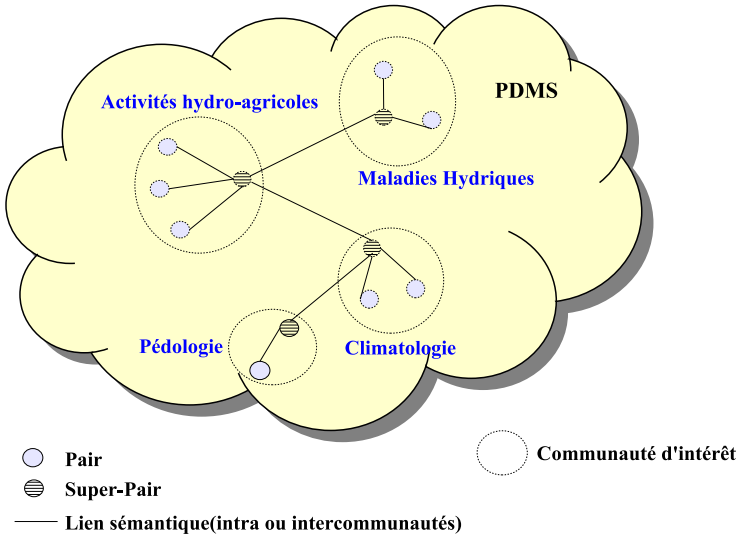


Figure 3.2 – Réseau Super-pair sémantique organisé par thèmes.

Les paragraphes suivants présentent les architectures fonctionnelles des différents types de nœuds.

3.3.2 Architecture d'un pair

Dans ce paragraphe, nous présentons l'architecture logique d'un pair en accord avec notre contexte et la topologie de SenPeer. Nous supposons que tous les pairs ont des architectures identiques, c'est-à-dire chaque nœud se conforme à l'architecture d'un pair SenPeer. Cependant, les capacités de stockage, de calcul et les bandes passantes peuvent différer d'un pair à l'autre. Les pairs sont une catégorie de nœuds avec un minimum de fonctionnalités. Ils ont la capacité de répondre à des requêtes sur leurs données et de recevoir les résultats de leurs requêtes, mais ne font pas de tâche d'indexation ni de traitement de requêtes distribuées. L'architecture générale d'un pair, matérialisant son interaction avec le reste du PDMS, est décrite dans la figure 3.3. La structure d'un pair est composée de trois couches, chaque couche de niveau supérieur offrant ses fonctionnalités en se basant sur celles offertes par une couche de niveau inférieur : *couche Application*, *couche Service*, *couche Représentation*.

En particulier, la *couche Représentation* fournit un moyen de gérer le schéma local du pair ainsi que sa représentation interne $sGraph$. Un adaptateur est utilisé pour convertir le schéma source dans un tel format. La *couche Service* contient le gestionnaire de requêtes qui est le principal service qu'un pair peut fournir au réseau. La *couche Application* contient une Interface Graphique Utilisateur (GUI) permettant à l'utilisateur de formuler des requêtes, mais aussi de consulter les résultats de ses requêtes.

Chaque pair dispose essentiellement des composantes suivantes réparties entre les trois niveaux : (figure 3.3) :

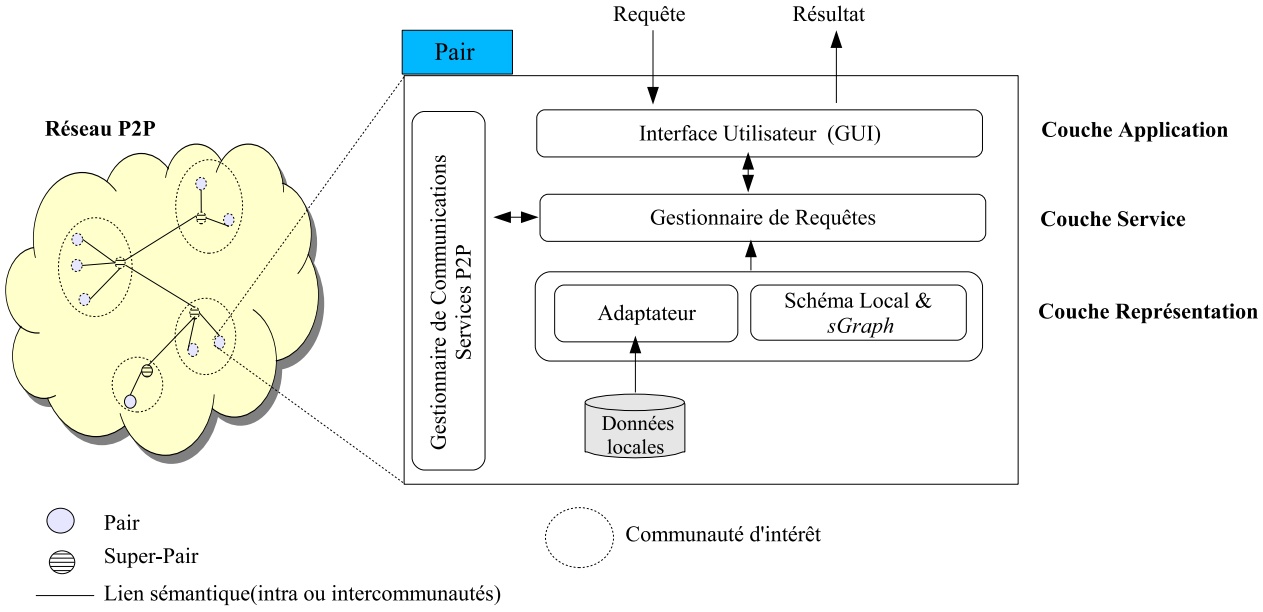


Figure 3.3 – Architecture d'un pair.

Source de données : Chaque pair dispose d'un système permettant de gérer ses données qui peuvent être dans la liste suivante : bases de données relationnelle, document XML ou RDF, etc. , et d'un langage d'interrogation (SQL, XQuery, etc.) en rapport avec son modèle de données. Ceci l'autorise à fonctionner seul indépendamment des autres pairs.

Adaptateur : Les pairs étant hétérogènes en termes de modèle de données et de langages de requêtes, les adaptateurs leur permettent de participer au réseau de partage de données en réécrivant les

requêtes du langage de requête interne vers les langages d'interrogation des pairs et vice-versa. Ils aident aussi à la découverte des correspondances en transformant le schéma local vers le format d'échange de schémas $sGraph$.

Réseau sémantique local ($sGraph$) : Les données publiées par le pair sont abstraites sous forme d'un réseau sémantique ($sGraph$) avec une annotation des nœuds par un ensemble de mots-clés issus des schémas et ajoutés par les concepteurs de ces derniers. Ce modèle interne constitue la représentation sémantique du contenu du pair. Il permet de faciliter l'échange des schémas entre pairs et a pour but de venir à bout de l'hétérogénéité syntaxique des pairs pour faciliter la découverte des correspondances sémantiques dans une communauté mais aussi entre communautés.

Gestionnaire de Requêtes : Ce module permet d'exprimer des requêtes locales ou globales. Les requêtes locales sont exécutées comme dans une base de données traditionnelle et les requêtes distantes routées vers le *parrain* de la communauté sémantique du pair pour une diffusion à la communauté ou aux communautés *amies*.

Interface graphique utilisateur : L'interface utilisateur permet à un pair d'importer des données, mais aussi de formuler plus facilement des requêtes locales sur ses données ou globales dans l'ensemble du réseau. Nous supposons que l'utilisateur n'a pas connaissance des schémas des pairs distants et qu'il formule ses requêtes sur son schéma local.

Gestionnaire de communication : La communication entre les pairs du système est assurée par le projet *Open Source* JXTA de Sun[55]. JXTA définit un réseau générique permettant de construire une variété de réseaux Pair-à-Pair tout en étant indépendant de la plate-forme, des langages de programmation (C ou Java) des systèmes (Microsoft Windows, Unix), des définitions de services (RMI, WSDL) et des protocoles réseaux (TCP/IP ou *Bluetooth*).

3.3.3 Architecture d'un super-pair

Comme pour les pairs, nous admettons que les super-pairs ont des architectures identiques, c'est-à-dire chaque nœud super-pair se conforme à l'architecture d'un super-pair SenPeer. Ils peuvent être hétérogènes en termes de capacité de calcul, bande passante, etc. L'architecture générale d'un super-pair, montrant son interaction avec le reste du PDMS, est décrite dans la figure 3.4.

Chaque super-pair contient les composantes suivantes :

Réseau Sémantique de la communauté (Schéma suggéré) : C'est un graphe (appelé $sGraph$) reflétant la structuration des données de la communauté dont le super-pair est responsable. Ce schéma est suggéré par le super-pair, mais les membres de la communauté ont la liberté de décrire leurs données avec des vocabulaires différents de celui suggéré pour la communauté.

Gestionnaire de Correspondances : Il est chargé de prendre les modèles internes des pairs de la communauté et ceux des autres super-pairs afin de générer les matrices de correspondance établissant les liens sémantiques entre les éléments de ces réseaux sémantiques et celui du super-pair.

Matrices de Correspondance : Elles stockent les correspondances trouvées par le gestionnaire de correspondances et sont maintenues par ce dernier. Elles sont de deux types : Super-pair/Super-pair ($\mathcal{M}_{SP/SP}$) contenant les correspondances entres les super-pairs responsables de deux communautés données et Super-pair/Pair ($\mathcal{M}_{SP/P}$) contenant les correspondances entre un super-pair et les pairs de sa communauté.

Index communautés : Cette composante contient les informations sur les pairs de la communauté locale mais aussi celles sur les super-pairs supervisant les communautés distantes qui sont sémantiquement liées. Ces informations sont, par exemple, les adresses IP, les bandes passantes,

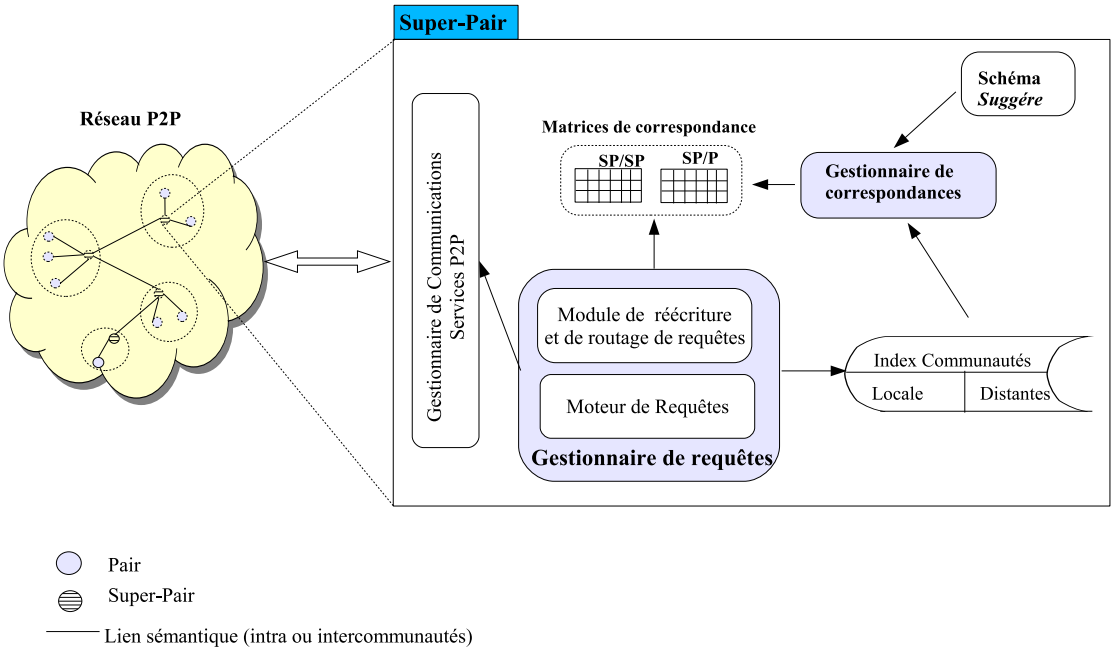


Figure 3.4 – Architecture d'un super-pair.

les ID et les expertises de ces (super-)pairs. Les expertises constituent des résumés succincts des schémas des pairs (c.f. paragraphe 3.7) et elles sont stockées dans deux types de tables : Super-pair/Super-pair ($E_{SP/SP}$) pour les expertises des super-pairs liés et Super-pair/Pair ($E_{SP/P}$) pour les expertises des pairs de la communauté. Ces expertises sont utilisées pour le routage efficace des requêtes vers les (super-)pairs pertinents.

Le Gestionnaire de Requêtes : Il contient le *Module de réécriture et de routage* de requêtes et le *Moteur de Requêtes*. La première composante est chargée de réécrire les requêtes et de les router vers les pairs et super-pairs concernés en accord avec les matrices de correspondance et les expertises des (super-)pairs connus. Étant donné une requête arrivant au super-pair, il génère les sous-requêtes sous forme de graphe pour les Adaptateurs des pairs, puis la fait suivre aux super-pairs liés. Le second est chargé de l'exécution des requêtes. Il définit les plans d'exécution et les optimise avant de superviser leur exécution dans l'ensemble du réseau.

Module de communication : Comme pour le pair, la communication est assurée par JXTA de Sun[55].

3.3.4 Communication entre (super-)pairs

Pour échanger les données entre (super-)pairs, SenPeer utilise la notion de *pipe* ou *canaux de communication* virtuels introduite dans plusieurs projets [100][55]. Un canal de communication est rattaché à un port TCP d'un pair et peut être mis à la disposition des autres pairs du réseau. Ces canaux sont *unidirectionnels*, en ce sens que les données transitent seulement dans un sens. Cependant, un canal *bidirectionnel* peut être implémenté en utilisant deux canaux *unidirectionnels* Dans notre PDMS nous définissons des canaux bidirectionnels entre le (super-)pair actuel et les autres (super-)pairs devant communiquer avec lui. Ces canaux permettent l'établissement de connections physiques entre les nœuds.

La figure 3.5 indique le processus d'établissement de canaux *bidirectionnels* entre nœuds. Le (super-)pair déclare sa promptitude à servir les autres (super-pairs) en créant un *canal d'entrée* et en avisant les (super-)pairs concernés. Cette information se fait sous la forme d'un message de *publication de canal* contenant l'ID du canal ainsi que d'autres informations supplémentaires telles que sa bande passante, son adresse IP, etc. A la réception de cette publication de canal, chacun de ces (super-)pairs intéressés par le dialogue avec le (super-)pair actuel envoie une demande de *raccordement de canal* à ce dernier. En cas d'acceptation, le (super-)pair, qui est par ailleurs le superviseur du canal, puisqu'ayant initié sa création, envoie un message d'*acceptation de raccordement de canal*. Quand l'un des (super-)pairs reçoit une réponse de ce type il :

(i) crée un *canal de sortie* pour se connecter au (super-)pair superviseur, puis

(ii) crée *un canal d'entrée* et envoie la *publication de ce canal* (en utilisant son canal de sortie) au (super-)pair pour être en mesure de recevoir des messages ultérieurs du (super-)pair de manière asynchrone.

De cette façon, nous avons des canaux bidirectionnels entre (super-)pairs désireux de partager leurs données. Notons que le canal d'entrée d'un super-pair sert de point d'entrée commun à tous les pairs qui lui sont affiliés et qu'il est utilisé pour l'envoi de requêtes au super-pair. D'autre part, les canaux d'entrée des pairs sont spécifiques à leurs propriétaires.

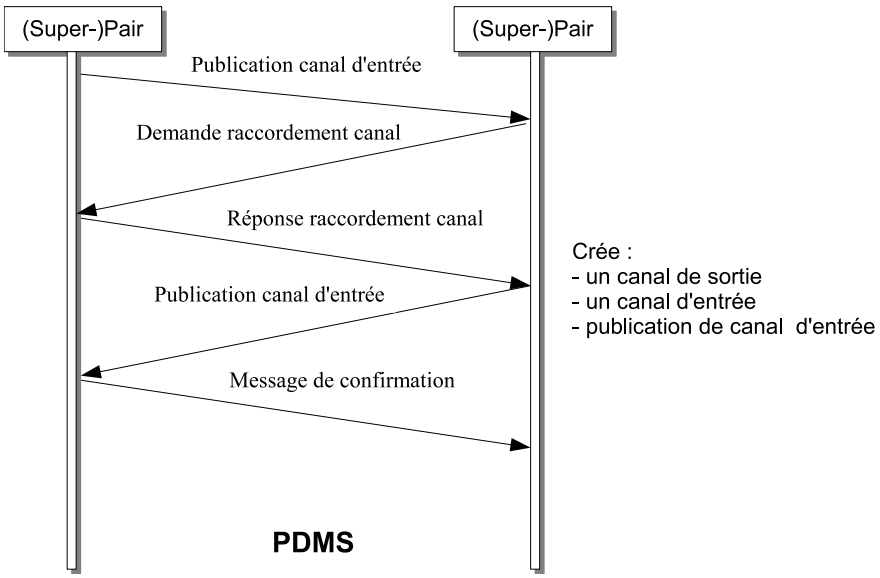


Figure 3.5 – Procédure de création d'un canal de communication bidirectionnel

3.4 Le problème de la médiation

La médiation a pour but principal de faciliter le partage de données entre pairs. Les données partagées étant réparties entre différents acteurs, ces derniers peuvent les décrire avec leur propres vocabulaires, ce qui résulte donc en une hétérogénéité sémantique. La réconciliation des ces vocabulaires passe par l'établissement d'un ensemble de correspondances sémantiques entre les schémas des données, correspondances qui seront ensuite utilisées pour la réécriture des requêtes. La découverte des

correspondances présente des défis pour plusieurs raisons. Premièrement, des schémas pour des contenus identiques peuvent avoir des différences structurelles et de nomenclature. Il peuvent utiliser des mots similaires mais avec des sens différents. D'autre part, ils peuvent être exprimés dans différents modèles de données.

Dans SenPeer, nous distinguons deux niveaux de réconciliation, tous pris en charge par les gestionnaires de correspondances des *parrains* super-pairs :

- Médiation entre les pairs d'une communauté, car les schémas de ces derniers ont été conçus par des experts différents. Dans ce cas, les correspondances sémantiques trouvées sont stockées dans la matrice de correspondance SP/P.
- Médiation entre super-pairs, puisque les communautés ne sont pas disjointes. Les correspondances trouvées sont stockées dans la matrice de correspondance SPS/P.

L'organisation des pairs en communautés sémantiques permet de ne pas établir des processus de découverte de correspondances deux-à-deux entre ces derniers, puisque cette solution requiert un effort quadratique en accord avec la taille du réseau.

Par ailleurs, la diversité des données publiées en termes de modèles de données constitue une difficulté supplémentaire pour la génération des correspondances. En effet, chaque pair participant au partage fournit des données de son domaine pouvant se conformer à un modèle de données quelconque. D'autre part, aucun des modèles de données considérés n'est assez expressif pour représenter les autres.

De la même façon que P. Berstein[16], nous pensons qu'une représentation pivot interne indépendante des modèles des sources de données des pairs est nécessaire pour faciliter la découverte des correspondances sémantiques. Cette étape de traduction devient un pré-requis indispensable à l'intégration et elle est faite à part par un adaptateur lié au modèle de données du pair en question. Les schémas des sources sont représentés dans un formalisme interne sous forme de réseau sémantique que nous appelons $sGraph$ (*semantic Graph*) tout en préservant les relations structurelles entre les objets spécifiques dans les langages des schémas. De plus, cette structure est enrichie sémantiquement grâce à un ensemble de mots-clés ajoutés à la conception des schémas. L'enrichissement sémantique est typiquement un processus de décision humaine faite à la conception des schémas et avant le processus automatique de découverte des correspondances sémantiques. Cet enrichissement fournit plus d'informations sur les éléments des schémas eux-mêmes. La figure 3.6 décrit l'architecture de médiation à deux niveaux.

Nous présentons dans la partie suivante le modèle pivot interne qui a pour but de venir à bout de l'hétérogénéité des modèles de données.

3.5 Structure du modèle interne

Le modèle interne est un formalisme permettant de supporter la diversité des modèles de données et de faciliter le processus de découverte de correspondances sémantiques. L'objectif n'est pas de mettre en place un modèle de données commun qui puisse subsumer tous les autres, mais plutôt un modèle qui puisse représenter les concepts sémantiques nécessaires des autres modèles pour venir à bout, et assez rapidement, de la diversité de vocabulaire. D'autre part, plus le modèle est riche en concepts, plus le processus d'intégration sera complexe puisqu'il devra résoudre les nombreuses divergences dues aux différents choix de modélisation faits par les différents concepteurs. Pour rendre l'intégration de données simple, l'alternative est de choisir un modèle de données avec un minimum de sémantique et dans lequel la représentation des données est assurée par des faits élémentaires pour lesquels il n'y a pas d'alternative de modélisation. Nous préférons une approche orientée-objets qui peut fournir, en plus, des méthodes pour implémenter des règles de traduction spécifiques.

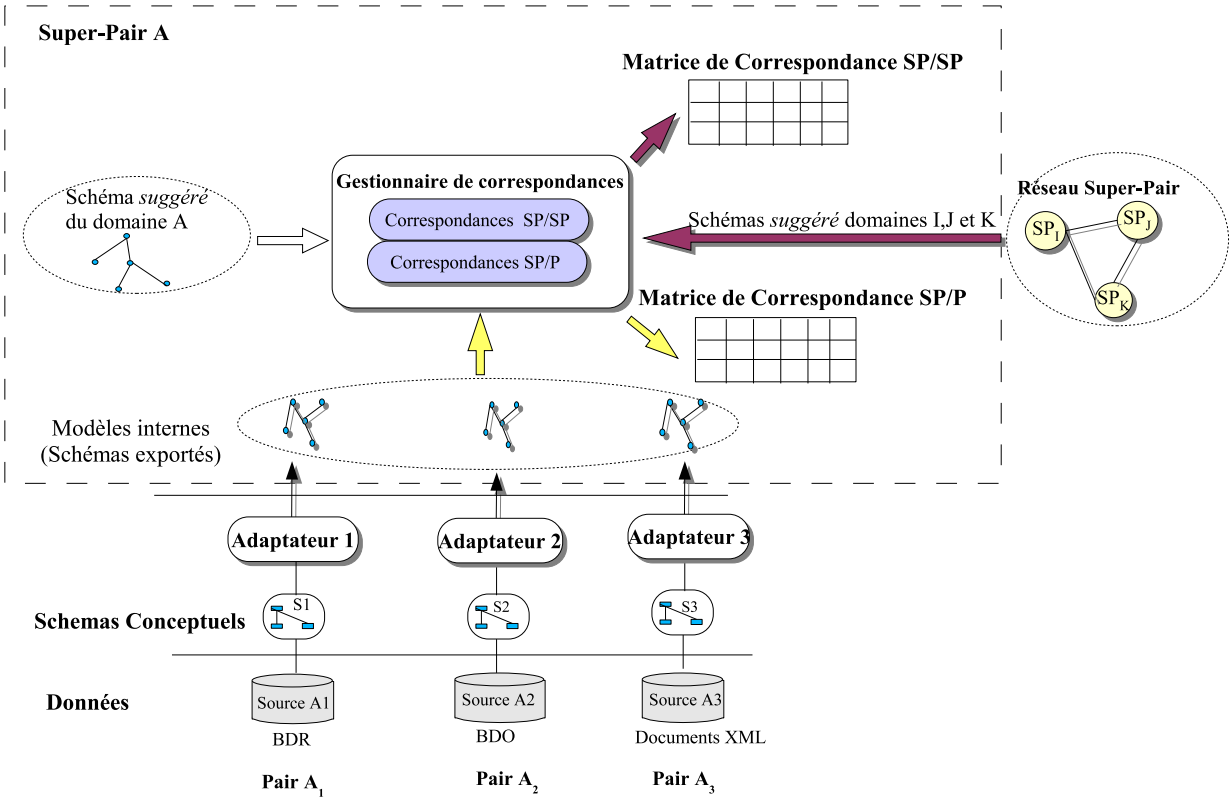


Figure 3.6 – Une architecture de médiation à deux niveaux.

Dans SenPeer, chaque pair exporte son schéma sous un format indépendant de son modèle de données et appelé $sGraph$. Un $sGraph$ est un graphe orienté et étiqueté dont les nœuds correspondent aux éléments dans les schémas sources (une table ou un attribut dans un schéma relationnel, un élément ou un attribut dans un schéma XML, une classe ou une propriété RDF, des définitions de types, etc.) et dont les arcs sont pris dans l'ensemble des relations sémantiques possibles entre ces éléments dans leurs schémas.

Définition 3.1. Soit $S = \{s_1, \ldots, s_p\}$ le schéma d'un pair dont les éléments s_i peuvent être des tables, des colonnes, des éléments ou des attributs XML. Nous définissons le s$Graph$ associé au schéma S comme un graphe orienté $\mathcal{G} = (N_S, E_S)$ avec $N_S = \{n_1, \ldots, n_p\}$ un ensemble fini de nœud et $E_S = \{e_1, \ldots, s_k\}$ un ensemble fini d'arcs. Chaque nœud $n \in N_S$ est étiqueté avec un élément de schéma tandis que chaque arc $e \in E_S$ représente une relation possible entre deux éléments de schémas. Les ensembles N_S et E_S sont définis formellement comme suit :

$N_S = \{n_1, \ldots, n_p\}$ ensemble des nœuds correspondant aux éléments figurant dans les schémas. Formellement l'étiquette d'un nœud $n_i \in N_S$ est définie par la fonction η qui fait correspondre à n_i une chaîne de caractères $s_i \in S$:

$$\begin{array}{rccl} \eta: & N_S & \rightarrow & S \\ & n_i & \rightarrow & s_i = \eta(n_i). \end{array}$$

$E_S = \{e_1, \ldots, e_k\}$ ensemble d'arcs. L'étiquette d'un arc $e_i = (n_1, n_2) \in E_S$ est donnée par la fonction ℓ qui fait correspondre e_i à une chaîne de caractères r_i appartenant à un ensemble $\Re$ de relation

sémantiques possibles entre éléments des schémas sources comme suit :

$$\begin{array}{rrcl} \ell: & E_S \subseteq N_S \times N_S & \rightarrow & \Re \\ & e_i & \rightarrow & r_i = \ell(e_i). \end{array}$$

Les relations sémantiques dans l'ensemble $\Re$ sont définies dans le paragraphe 3.5.2.

Les chaînes s_i et r_i sont des concepts non nuls. En résumé, un $sGraph$ est un ensemble $\mathbb{SG} = \{\mathcal{G} = (N_S, E_S), S, \Re, \eta, \ell\}$.

3.5.1 Etiquettes des nœuds

Les étiquettes des nœuds proviennent d'un ensemble de constructions provenant du langage naturel et ayant une interprétation sémantique. L'ensemble des étiquettes des nœuds est disjoint de celui des arcs. En principe, les étiquettes des nœuds peuvent être n'importe quelle chaîne de caractère. Cependant pour permettre l'interprétation sémantique nous supposons que la plupart des étiquettes sont des expressions linguistiques dans un langage naturel. Ce langage dépend du contexte. De plus nous supposons que ce sous-ensemble est pris d'un vocabulaire de domaine spécifique (Agriculture, climatologie, santé, etc.). Ces étiquettes font partie d'une variété d'expression linguistiques :

- étiquettes simples (composées d'un mot)
  nom commun, noms propres, etc.
- phrase nominale (PN)
  « taux sel », « humidité relative »
- phrase prépositionnelle (PP)
  « par inondation », « au seuil »
- phrase verbale
  « fermer barrage »
- De plus, ces expressions peuvent être combinées avec des séparateurs
  « culture-irriguée », « cumul_précipitations »

La grammaire suivante permet de définir formellement les étiquettes des nœuds :

L_N	:=	$(S\,(C\,S)^*)$
S	:=	$(PN \mid PP \mid PV \mid$ « » $\mid$ « - » $\mid$ « _ » $\mid$ « , »)
PV	:=	$(V\,(PN \mid PP)^?)$
PN	:=	$((PNPP) \mid (D^?\,N^*))$
PP	:=	$(P\,PN)$
N	:=	$nom\ commun$ ou $nom\ propre$
D	:=	$article$
P	:=	$préposition$
V	:=	$verbe$
C	:=	$conjonction$

Table 3.1 – Grammaire pour les étiquettes des nœuds.

3.5.2 Etiquettes des relations

Dans un $sGraph$, les relations utilisées sont binaires, orientées et typées. Les relations réflexives ne sont pas autorisées. Elles constituent toutes des chaînes de caractères atomiques. Ces relations sont

prises d'entre celles possibles entre les éléments dans leurs modèles sémantiques d'origine et nous les avons identifiées comme étant importantes pour la découverte des correspondances sémantiques. Nous en donnons une description sémantique ci-après :

- $isA(n_1, n_2)$

  n_1 est une spécialisation n_2. Cette relation va d'un concept spécifique à un concept plus général. Elle est transitive et asymétrique et définit une structure hiérarchique entre les éléments qu'elle relie.

- $contains(n_1, n_2)$

  n_1 est un conteneur pour n_2. Intuitivement, un contenu ne peut exister seul car il dépend de l'existence de son conteneur. Par exemple, une base de données contient des tables qui contiennent elles mêmes des colonnes. Cette relation a pour propriété la propagation de la suppression.

- $partOf(n_1, n_2)$

  n_1 est une sous-composante de n_2. Elle permet de regrouper des éléments. Cette relation est plus faible que $contains$ car elle ne propage pas les suppressions. Par exemple, une clé concaténée agrège les colonnes d'une table.

- $typeOf(n_1, n_2)$

  n_1 est de type n_2. Cette relation permet de relier un élément de schéma à un autre nœud représentant son méta-type.

- $associates(n_1, n_2)$

  n_1 est associé à n_2, mais rien n'est dit sur la sémantique de cette relation. C'est la relation la plus faible qui peut être exprimée. Elle n'a pas de contrainte sémantique spéciale. Par exemple, une table est associée à une autre.

- $dataType(n_1, n_2)$

  n_1 a pour type de données n_2. Si l'élément contient des données, la relation $dataType$ permet de lui associer un type indiquant le format de la donnée stockée. Les types de données sont les mêmes que ceux des schémas d'origine : int, $float$, $string$, etc.

- $sourceType(n_1, n_2)$

  n_1 a pour modèle de données n_2. Cette relation permet d'indiquer le modèle de données originel du $sGraph$ actuel.

- $hasId(n_1, n_2)$

  n_1 a pour identificateur n_2. Elle permet d'associer un élément à un identificateur unique.

- $references(n_1, n_2)$

  n_1 fait référence à n_2. Elle constitue un moyen d'associer des éléments entre eux à l'instar des clés d'enregistrement ou des IDREF XML. Par exemple, une colonne clé étrangère fait référence à une colonne clé primaire.

3.5.3 Enrichissement sémantique

Pour capturer la sémantique des termes utilisés comme noms des nœuds dans un $sGraph$, nous associons à chaque nœud n_i un ensemble de mots-clés $syn(n_i)$. Cet ensemble de mots-clés constitue essentiellement des synonymes issus des schémas et ajoutés par leurs concepteurs et destinés à guider le processus de découverte des correspondances sémantiques. Ces mots-clés permettent donc de fournir

plus d'informations linguistiques sur l'élément correspondant. L'utilisation d'ensemble de mots en complément du nom du nœud permet de tenir compte de la polysémie et de la synonymie dans le processus de l'interprétation sémantique des étiquettes des nœuds. Par exemple, le mot *culture* dénote plus d'un concept (par exemple le traitement du sol en vue de la production agricole, un ensemble de connaissances adoptées par un groupe social ou une culture bactériologique) tandis que l'ensemble de synonymes constitué par *culture*, *instruction* et *savoir* identifie un concept unique, celui des connaissances et valeurs partagées par une société.

Certains nœuds, comme les nœuds représentant des types, ne disposent pas de mots-clés associés puisque leur similarité est plus facile à calculer et elle est prédéfinie. Notons aussi que le nom du nœud fait partie de l'ensemble de ses synonymes. La nomenclature des synonymes suit la grammaire utilisée pour définir les étiquettes des nœuds.

En revenant sur l'exemple de la vallée du fleuve sénégal, la figure 3.7 constitue une partie d'un modèle interne (à l'origine une base de données relationnelle) exprimé dans le formalisme $sGraph$ et représentant une source de données sur les types de cultures pratiquées sur les sols de la vallée. Par souci de lisibilité, nous avons volontairement omis les mots-clés associés à certains nœuds. La racine est indiquée par le nœud avec le contour épais. La base de données, les noms des tables et des colonnes sont représentés par des cercles non remplis, tandis que les nœuds remplis correspondent à des types d'éléments (tables, colonnes, etc.) ou à des types de données ($integer$, $string$, $float$, etc.)

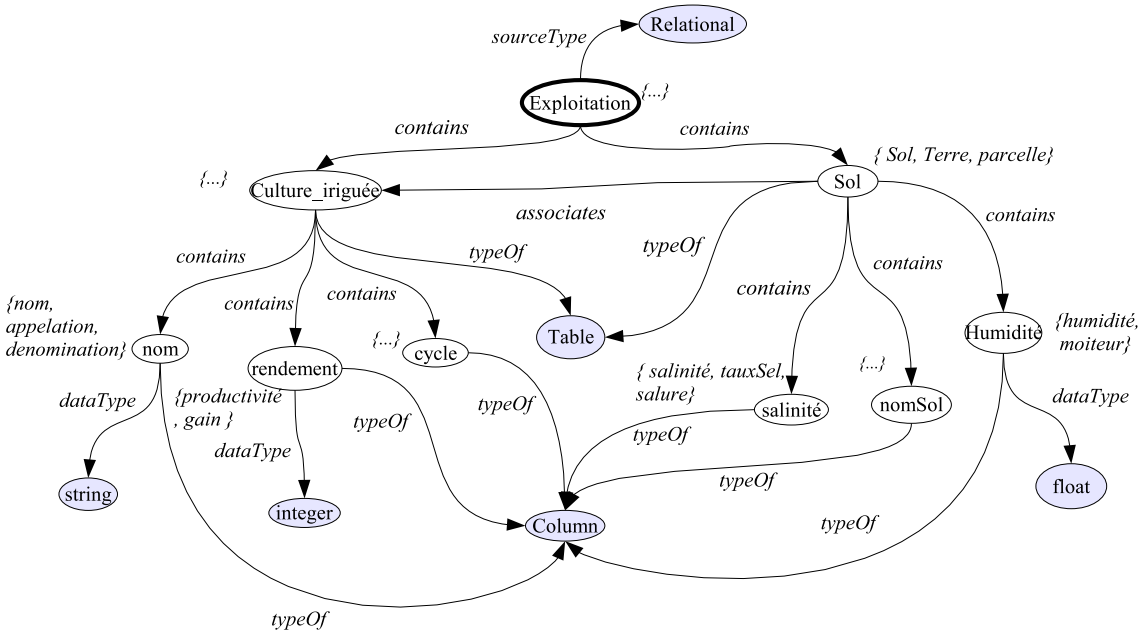


Figure 3.7 – $sGraph$ partiel sur les cultures pratiquées sur les sols du bassin du fleuve.

La structure d'un $sGraph$ quelconque peut être décrite formellement, dans un formalisme orienté objet avec la syntaxe BNF (figure 3.8).

Nous nous basons sur la technologie XML pour implémenter les $sGraphs$. Une syntaxe XML est définie pour décrire et stocker les $sGraphs$ en mémoire secondaire, par exemple disque (voir Annexes). Un parseur XML est ensuite utilisé pour les charger en mémoire centrale.

$< SNode >$ ::= class **{**
$name$: <*word*>
$synonym$: {}|{<*syn-set*>}
$typeOf$: <*typeNode*>
$sourceType$: <*sourceTypeNode*>
$dataType$: <*dataTypeNode*>
isA : <*isA*>
$contains$: <*contains*>
$partOf$: <*partOf*>
$associates$: <*associates*>
$hasId$: <*hasId*>
$references$: <*references*> **}**
<*syn-set*> ::= <*word*> | <*word*>,<*synset*>
<*isA*> ::= {} | {$< SNode >$}
<*typeNode*> ::={} | $Table$ | $Column$ | $Element$ | $Attribute$ | $Class$ | $ComplexeType$ | PK | FK | ID | $IDREF$
<*dataTypeNode*> ::={} | $Integer$ | $Float$ | $String$
<*sourceTypeNode*>::={} | $RelationalDB$ | $ObjectDB$ | $XMLDocument$
<*contains*> ::={} | {$< SNode >$}
<*partOf*> ::={} | {$< SNode >$}
<*associates*>::= {} | {$< SNode >$}
<*hasId*> ::= {} | {$< SNode >$}
<*references*>::= {} | {$< SNode >$}

Figure 3.8 – Définition de la classe représentant un nœud du modèle interne $sGraph$.

3.5.4 Navigation dans un $sGraph$

Dans le but de naviguer dans un $sGraph$, nous définissons un ensemble d'opérations. Nous avons besoin de ces opérations pour identifier les concepts représentant les nœuds nous intéressant pour la médiation sémantique et le résumé des schémas sous forme d'expertises. Nous introduisons d'abord les notation suivantes :

- $Node(c)$ est le nœud étiqueté par le concept c ;
- $Child(n)$ dénote un fils du nœud n ;
- $Parent(n)$ représente un nœud parent de n ;

Soit $n_1 = Node(c_1)$ et $n_2 = Node(c_2)$, les opérations sont formulées comme suit :

- $inComing$

$$n_2 \in inComing(n_1, r) \text{ ssi } n_2 = Parent(n_1) \wedge \ell[(n_2, n_1)] = r. \tag{3.1}$$

Informellement, cette relation retourne les nœuds parents d'un nœud n en suivant un arc de type r.

- $outGoing$

$$n_2 \in outGoing(n_1, r) \text{ ssi } n_2 = Child(n_1) \wedge \ell[(n_1, n_2)] = r. \tag{3.2}$$

Cette relation retourne les nœuds fils d'un nœud n en suivant un arc de type r.

- syn

$$syn(n) = synonym(n). \tag{3.3}$$

Cette relation retourne l'ensemble des synonymes du nœud n.

- $InternalNodes$

$$InternalNodes(SG) = \{n \in SG | \exists\, n' \in SG \wedge \exists\, r \in \Re \wedge (n' \in outGoing(n, r))\} \tag{3.4}$$

Intuitivement, cette relation retourne l'ensemble des nœuds internes du $sGraph$ (autres que les nœuds types et types de données).

A partir de ces opérations de navigation, nous définissons les prédicats suivants :

$$Type(n_1, n_2) = \begin{cases} vrai & \text{Si } outGoing(n_1, \texttt{"typeOf"}) = n_2 \wedge \\ & n_1 \in inComing(n_2, \texttt{"typeOf"}) \\ faux & \text{Sinon} \end{cases} \quad (3.5)$$

$$Contains(n_1, n_2) = \begin{cases} vrai & \text{Si } n_2 \in outGoing(n_1, \texttt{"contains"}) \wedge \\ & inComing(n_2, \texttt{"contains"}) = n_1 \\ faux & \text{Sinon} \end{cases} \quad (3.6)$$

$$Associates(n_1, n_2) = \begin{cases} vrai & \text{Si } n_2 \in outGoing(n_1, \texttt{"associates"}) \wedge \\ & n_1 \in inComing(n_2, \texttt{"associates"}) \\ faux & \text{Sinon} \end{cases} \quad (3.7)$$

3.6 Réconciliation sémantique

La réconciliation sémantique des schémas passe par la découverte d'un ensemble de correspondances sémantiques établies grâce à des mesures de similarité sémantique entre les éléments des schémas. En l'absence des instances de données, les noms des éléments des schémas, leurs descriptions, leur relations, les contraintes, etc., sont probablement les sources d'informations les plus utiles pour l'appariement des schémas. La découverte des correspondances peut être considérée comme un problème de recherche dans un espace très large, c'est-à-dire avec un nombre de correspondances candidates potentiellement élevé. Pour explorer l'espace de recherche de façon efficace, nous combinons un ensemble de techniques d'appariement. Chaque technique considère un sous-ensemble significatif de l'espace de recherche : noms et synonymes des éléments, types de données des éléments, relations sémantiques entre éléments. Les mesures de similarité tiennent compte des aspects linguistiques et structurelles des éléments. Les résultats partiels de chaque approche sont combinées pour générer une valeur ou score de similarité globale qui se présente comme une somme pondérée de chacun des scores partiels.

Le problème de la découverte de correspondance sémantiques entre deux $sGraph$ peut être formulé comme suit :

Définition 3.2. [***Découverte de correspondances***]
Étant donné deux $sGraph$ S et T et des informations auxiliaires (thésaurus), le processus de découverte de correspondances consiste à calculer $|S|\times|T|$ éléments de correspondances $< ID_{ij}, n_i^S, n_j^T, R, \gamma_{ij} >$ qui se passent de la validation de l'utilisateur, avec $n_i^S \in S$ $(1 \leq i \leq |S|)$, $n_j^T \in T (1 \leq j \leq |T|)$, R la relation sémantique entre les concepts représentés par les deux nœuds et γ_{ij} le degré de similarité entre les nœuds considérés. La valeur γ_{ij} peut être vue comme la confiance accordée à la correspondance.

Les correspondances sémantiques sont définies après l'analyse de la similarité (dans l'intervalle $[0, 1]$, 0 correspondant à une dissimilitude forte et 1 à une similarité parfaite) des nœuds des $sGraph$ comparés. Cette valeur de similarité est un indicateur sur la plausibilité de la correspondance. L'affinité sémantique entre les concepts est établie si leur similarité est supérieure à un seuil minimum de similarité.

Les paragraphes suivants introduisent les mesures de similarité permettant d'établir les correspondances entre les éléments de deux modèles représentés avec le formalisme $sGraph$.

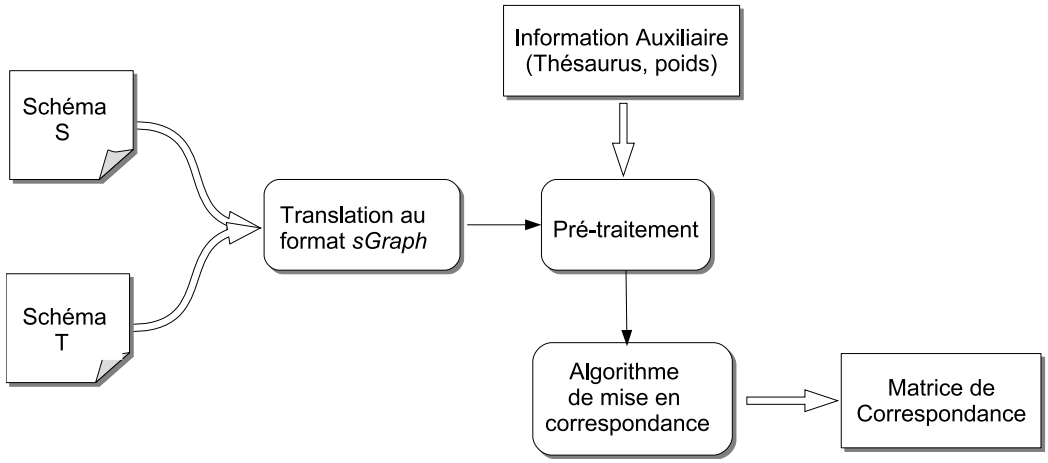


Figure 3.9 – Processus de découverte de correspondances entre schémas.

3.6.1 Similarité globale

Nous utilisons un processus de découverte de correspondances basé sur le modèle pivot. Notre méthode est basée sur les schémas et non sur les instances. La génération des correspondances entre éléments passe par la définition d'une mesure de similarité entre ces derniers. Nous partageons les approches générales avec les algorithmes classiques[97][25][31] de découverte de correspondances telles que l'évaluation de la qualité de l'appariement dans l'intervalle $[0, 1]$, l'appariement des structures en se basant sur le voisinage local. Notre mesure de similarité est donc composite et dépend des affinités linguistique et structurelle des éléments figurant dans les schémas. La similarité entre deux nœuds appartenant à deux $sGraph$ différents est fonction de :

- leurs descriptions sémantiques avec des mots-clés et de leurs types ;
- leurs relations sémantiques avec les autres concepts (voisinage).

Définition 3.3. [***similarité globale***]
La fonction de similarité $Sim(n_i^S, n_j^T)$ de deux nœuds n_i^S et n_j^T appartenant à deux $sGraph$ S et T est une somme pondérée des similarités linguistique $S_l(n_i^S, n_j^T)$ et de voisinage $S_v(n_i^S, n_j^T)$ de ces deux nœuds, soit :

$$Sim(n_i^S, n_j^T) = \lambda_l.S_l(n_i^S, n_j^T) + \lambda_v.S_v(n_i^S, n_j^T) \in [0, 1] \quad \text{avec} \quad \lambda_l, \lambda_v \geq 0 \text{ et } \lambda_l + \lambda_v = 1. \quad (3.8)$$

Les coefficients λ_l et λ_v sont les poids associés respectivement aux synonymes et aux relations de voisinage des concepts. Ils permettent de normaliser la similarité globale.

3.6.2 Similarité linguistique

Les informations sur le nom d'un nœud et sur l'ensemble de ses synonymes fournissent un premier niveau de sémantique *évident* sur la sémantique possible du nœud considéré. La similarité linguistique $S_l(n_i^S, n_j^T)$ de deux éléments n_i^S et n_j^T permet d'évaluer l'affinité linguistique entre les deux éléments[25]. Elle est calculée automatiquement en se basant sur la connaissance lexicale : connaissance à propos des étiquettes des nœuds mais aussi de leurs ensembles de synonymes. Le but de la comparaison

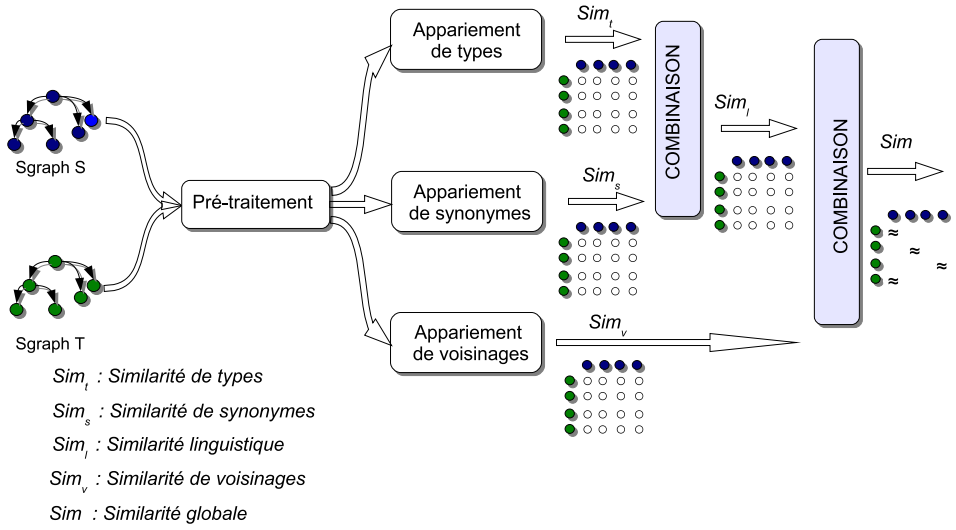


Figure 3.10 – Combinaison de plusieurs *apparieurs*.

des ensembles de synonymes est d'exploiter les ressemblances générales dans l'utilisation des mots et de détecter les mots équivalents qui vraisemblablement font référence au même concept. Par conséquent, les ensembles de synonymes peuvent être utilisés seulement pour détecter une équivalence linguistique entre les concepts représentés par les nœuds.

3.6.2.1 Similarité lexicale

La similarité lexicale entre deux mots Ts_i et Tt_j est basée sur la mesure de similarité lexicale SM proposée par [73], elle même basée sur la distance d_L de LEVENSHTEIN[64].

$$SM(Ts_i, Tt_j) = \max\left(0, \frac{min(|Ts_i|, |Tt_j|) - d_L(Ts_i, Tt_j)}{min(|Ts_i|, |Tt_j|)}\right) \in [0, 1]. \tag{3.9}$$

Cette fonction calcule la similarité des chaînes Ts_i et Tt_j en tenant compte du nombre d'actions atomiques (ajout, suppression de caractère, modification) nécessaires pour transformer l'une des chaînes de caractères en l'autre chaîne. Ce nombre d'actions atomiques est calculé grâce à la distance d_L de LEVENSHTEIN. La fonction SM est fonction du rapport entre le nombre de ces opérations d'édition et de la longueur de la plus courte des deux chaînes Ts_i et Tt_j à comparer. Cette fonction permet de normaliser la valeur de la distance de LEVENSHTEIN par le maximum entre la longueur des deux chaînes, c'est-à-dire, la fonction vaut 0 si les chaînes sont complètement différentes et 1 si elles ont exactement les mêmes.

Prenons par exemple les deux entrées lexicales $cultureirrigue$ et $culture\_irriguee$. Puisque $d_L(cultureirrigue, culture\_irriguee) = 1$ on a alors $SM(cultureirrigue, culture\_irriguee) = \frac{13}{14}$

D'autres fonctions classiques de calcul de similarité entre chaînes telles que la distance de Hamming, la distance $n\text{-}gram$, etc. sont applicables à la place de la fonction SM.

3.6.2.2 Similarité de deux synonymes

La similarité globale entre les deux ensembles de synonymes dépend de la comparaison lexicale entre les différents mots ou groupes de mots (un synonyme pouvant être un groupe de mots) de ces deux ensembles. Dans le cas où les synonymes sont des groupes de mots, la similarité entre deux synonymes s et t dépend alors des termes les composant. Pour ce faire, les synonymes sont décomposés en termes atomiques en utilisant les caractères spéciaux, les majuscules, etc. comme délimiteurs. Les termes représentant des articles, des prépositions, des conjonctions, etc. sont supprimés, car n'apportant pas d'information pertinente[31].

La similarité $ST(s,t)$ de deux synonymes (ensembles de termes) s et t est calculée comme une moyenne des meilleures valeurs de similarité de chaque terme source avec un terme cible. Formellement $ST(s,t)$ est définie comme suit :

$$ST(s,t) = \frac{\sum_{Ts_i \in s} \left[\max_{Tt_j \in t} SM(Ts_i, Tt_j)\right] + \sum_{Tt_j \in t} \left[\max_{Ts_i \in s} SM(Tt_j, Ts_i)\right]}{n+m}. \tag{3.10}$$

avec Ts_i, $1 \leq i \leq n$ un terme de s et $Tt_j, 1 \leq j \leq m$ un terme de t, n et m les nombres de termes de s et t respectivement.

3.6.2.3 Similarité d'ensembles de synonymes

La similarité $S_s(n_i^S, n_j^T)$ entre les ensembles de synonymes est basée sur la mesure proposée par TVERSKY[111]. Pour comparer deux ensembles d'éléments A et B, TVERSKY se base sur les intuitions suivantes :

- La similarité entre A et B dépend de ce qu'ils ont en commun. Plus ils ont de choses en commun, plus leur similarité sera élevée.
- La similarité entre A et B dépend de leurs différences. Plus ils ont de différences, plus leur similarité sera faible.

La mesure de TVERSKY est donnée par la formule ci-dessous dans laquelle $A \cap B$ représente l'intersection des ensembles A et B et $A \setminus B$ leur différence, $|A|$ le cardinal de A et α une valeur indiquant l'importance des caractéristiques communes et non communes.

$$S(a,b) = \frac{|A \cap B|}{|A \cap B| + \alpha|A \setminus B| + (1-\alpha)|B \setminus A|} \quad \text{avec} \quad 0 \leq \alpha \leq 1. \tag{3.11}$$

Cette mesure est pratique, mais telle qu'elle, elle n'est pas tout à fait adaptée à notre cas puisqu'elle ne prend en compte qu'un appariement exact des termes. Par exemple les termes $Culture\_irriguée$ et $CultureIrriguée$ sont considérés comme différents alors qu'ils sont, à notre avis, équivalents à une transformation près. Nous nous proposons de l'étendre en faisant un appariement flou des termes présents dans les ensembles de synonymes, plutôt qu'un appariement exact de ces termes.

Définition 3.4. [***Intersection et différence floues***]

Étant donné A et B deux ensembles d'entrées lexicales, nous définissons l'intersection floue $A \cap_f B$ et la différence floue $A -_f B$ de ces deux ensembles comme suit :

$$\begin{aligned} A \cap_f B &= \{a \in A / \max_{b \in B} SM(a,b) > \epsilon_{acc}\}. \\ A -_f B &= \{a \in A / \max_{b \in B} SM(a,b) \leq \epsilon_{acc}\}. \end{aligned} \tag{3.12}$$

ϵ_{acc} est le seuil au dessus duquel la similarité calculée est considérée comme acceptable.

Nous pouvons maintenant définir la similarité de deux ensemble de synonymes

Définition 3.5. [***Similarité de deux ensembles de synonymes***]
La similarité $S_s(n_i^S, n_j^T)$ entre deux ensembles de synonymes $syn(n_i^S)$ et $syn(n_j^T)$ de deux nœuds n_i^S et n_j^T s'exprime alors comme suit :

$$S_s(n_i^S, n_j^T) = \frac{|syn(n_i^S) \cap_f syn(n_j^T)|}{|syn(n_i^S) \cap_f syn(n_j^T)| + \alpha|syn(n_i^S) -_f syn(n_j^T)| + (1-\alpha)|syn(n_j^T) -_f syn(n_i^S)|} \in [0,1]. \tag{3.13}$$

3.6.2.4 Similarité de type

Le type de données renseigne sur le format des données correspondant aux nœuds. Le type de données fournit des pistes de correspondances en ce sens que des éléments de schémas similaires ont le plus souvent des types compatibles. En n'autorisant que des types de données compatibles et si ces types sont correctement affectés aux éléments, c'est-à-dire, s'ils sont le plus spécifique possible, ils peuvent réduire de façon drastique l'espace de recherche, c'est-à-dire, le nombre de correspondances candidates. Le seul problème qui pourrait surgir serait quand les types de données d'une source ne seraient pas assez spécifiques. Cette lacune peut néanmoins être comblée en tenant compte des similarités de synonymes et des similarités structurelles.

Les recommandations sur XML Schema fournissent 44 types de données différents dont 19 types primitifs et 25 types dérivés. C'est probablement l'ensemble de types de données idéal étant donné qu'il est assez raffiné dans le but de la découverte de correspondances. En effet, XML Schema permet la définition de types de données très spécifiques. Pour cela, la hiérarchie des types de données de XML Schema est utilisée dans le but de calculer les coefficients de compatibilité des types de données. Les types de données de XML Schema sont classifiés en plusieurs catégories (appelées types de données primitifs) incluant par exemple les types $Duration$, $Boolean$, $String$, $Decimal$, etc. Chaque catégorie a plusieurs types de données dérivés. Deux types de données sont considérés comme similaires s'ils appartiennent à la même catégorie de type de données et leur compatibilité de types de données dépend de leurs positions respectives dans la hiérarchie des types de données XML.

En nous basant sur la hiérarchie des types de données de XML schema[33], nous construisons une table de compatibilité[19][25][87] de type de données comme celle utilisée dans [71] et qui fournit un coefficient de similarité (dans l'intervalle $[0, 1]$) entre deux types de données. La table de compatibilité est construite pour 7 categories de types. Après cela, en comparant deux types de données, nous déterminons d'abord à quelle catégories ces types appartiennent et ensuite nous prenons la mesure de similarité dans la table de compatibilité des catégories.

La similarité de type de données est définie par le degré de préservation de l'information quand un nœud source est transformé en un nœud cible. Les similarités de types sont classifiées en quatre groupes en accord avec le degré de perte d'information :

- transformation équivalente ;
- transformation sans perte ;
- transformation avec perte ;
- transformation impossible.

La transformation équivalente bénéficie de la plus grande similarité tandis que celle impossible se voit affecter la similarité la plus faible.

Le tableau 3.2 définit les similarités de types de données pour 7 types de données représentatifs venant de 44 types définis par les spécifications de XML Schema.

Type source \ Type cible	$string$	$decimal$	$double$	$float$	$boolean$	$date$	$duration$
$string$	1.0	0.33	0.33	0.33	0.0	0.33	0.33
$decimal$	0.66	1.0	0.66	0.66	0.33	0.0	0.0
$double$	0.66	0.33	1.0	0.33	0.0	0.0	0.0
$float$	0.66	0.33	0.66	1.0	0.0	0.0	0.0
$boolean$	0.66	0.66	0.0	0.0	1.0	0.0	0.0
$date$	0.66	0.0	0.0	0.0	0.33	1.0	0.33
$duration$	0.66	0.0	0.0	0.0	0.0	0.33	1.0

Table 3.2 – Exemple de similarités de types de données.

3.6.2.5 Similarité linguistique

Définition 3.6. [***similarité linguistique***]
La similarité linguistique de deux nœuds n_i^S et n_j^T appartenant à deux $sGraph$ S et T constitue une somme pondérée de la similarité de leurs deux ensembles de synonymes $S_s(n_i^S, n_j^T)$ et de leur similarité de types $S_t(n_i^S, n_j^T)$ [71]:

$$S_l(n_i^S, n_j^T) = \omega_s.S_s(n_i^S, n_j^T) + \omega_t.S_t(n_i^S, n_j^T) \in [0,1] \quad \text{avec} \quad \omega_s, \omega_t \geq 0 \text{ et } \omega_s + \omega_t = 1. \tag{3.14}$$

La comparaison linguistique fournit un niveau d'assertion de similarité très basique, qui est une forme d'assertion de similarité peu concluante étant donné que les mots peuvent être différents tandis les nœuds sont sémantiquement liés. Les mots *rendement* et *productivité* en sont une illustration puisqu'ayant seulement peu de caractères en commun et par conséquent leur similarité syntaxique est très faible. De plus, même si les ensembles de synonymes sont explorés, les termes les composant peuvent avoir aussi, à leur tour, une similarité syntaxique très faible, ce qui résulte en une similarité linguistique faible. Cependant, la similarité sémantique de *rendement* et *productivité* est normalement très élevée. Inversement, les mots dans les ensembles de synonymes peuvent être les mêmes, tandis que les nœuds correspondants ne sont pas sémantiquement liés. La similarité de voisinage sémantique ou similarité structurelle introduite dans le paragraphe suivant vient en complément à celle dite linguistique.

3.6.3 Similarité de voisinage sémantique

La similarité globale de deux nœuds dépend aussi de leur contexte d'apparition dans les deux $sGraph$, donc de leur voisinage sémantique. Elle se base sur l'intuition selon laquelle des nœuds *similaires* sont liés à des nœuds *similaires*.

Définition 3.7. [***Voisinage sémantique***]
Soit $\mathcal{SG} = \{\mathcal{G} = (N_S, E_S), S, \Re, \eta, \ell\}$ un $sGraph$ et $n^S \in N_S$, le voisinage sémantique $V(n^S)$ de n^S est constitué de l'ensemble des nœuds ayant un lien sémantique direct avec n^S, soit :

$$V(n^S) = \{n_j^S \in InternalNodes(SG) / \exists\, r \in \Re \wedge r(n^S, n_j^S)\}. \tag{3.15}$$

$InternalNodes(SG)$ est constitué de l'ensemble des nœuds du $sGraph$ SG sauf les nœuds représentant des types. Le voisinage sémantique $V(n)$ d'un nœud est représenté à travers un *vecteur de voisinage*

$\overrightarrow{VV}(n) = (vv_1, \ldots, vv_k)$ avec $\forall\, 1 \leq i \leq k \quad vv_i = (n_i, r_p)$ avec n_i dénotant un nœud adjacent à n et $r_p \in \Re$ une relation sémantique entre n et n_i, c'est à dire $r_p(n_i, n)$ où $r_p(n, n_i)$

Par exemple, le vecteur de voisinage du nœud $culture\_irriguee$ est le suivant :

$$\overrightarrow{VV}(culture\_irriguee) = ((nom, contains), (rendement, contains), (cycle, contains), (sol, asscociates), (exploitation, contains)).$$

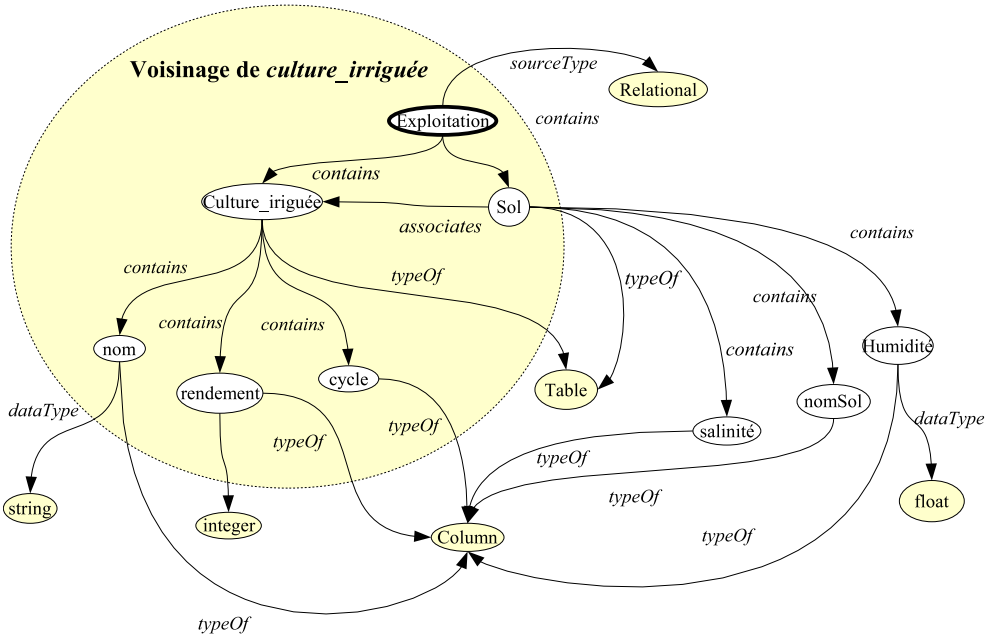


Figure 3.11 – Voisinage sémantique du nœud $culture\_irriguée$

La similarité de voisinage sémantique compare les nœuds dans les voisinages en se basant sur leurs similarité linguistiques. Elle dépend de la taille et de l'intersection floue de leurs voisinages. Elle est calculée en faisant un appariement flou des voisinages.

Définition 3.8. [***similarité de voisinage***] La similarité de voisinage de deux nœuds n_i^S et n_j^T appartenant à deux $sGraph$ S et T est définie comme suit :

$$S_v(n_i^S, n_j^T) = \frac{|\{x \in V(n_i^S) / \exists\, y \in V(n_j^T) \wedge Sim(x, y) > \epsilon_{acc}\}|}{|V(n_i^S) \cup V(n_j^T)|} \in [0, 1]. \tag{3.16}$$

Notons que les similarités de voisinages ne sont calculées qu'entre les nœuds internes des $sGraph$, car celles entre les nœuds externes (qui sont les nœuds représentant les types) sont déjà prédéfinies dans une table.

3.6.4 Génération des matrices de correspondance

A chaque domaine sont associés deux types de matrices de correspondance localisées au niveau du super-pair.

- matrice de correspondance **Super-pair/Pair (SP/P)** : Pour chaque pair de la communauté le parrain super-pair maintient des matrices de correspondance de ce type contenant les correspondances entre le vocabulaire qu'il suggère pour la communauté et celui du pair.
- matrice de correspondance **Super-pair/Super-pair (SP/SP)** : Pour chaque Super-pair dans son voisinage sémantique, le super-pair dispose de matrices indiquant les correspondances entre son *schéma suggéré* et les *schémas suggérés* par ses alliés super-pairs.

Intuitivement, un schéma suggéré est une représentation intentionnelle des données fournies par une communauté tandis que les correspondances sémantiques spécifient comment une telle représentation intentionnelle est liée aux schémas des membres de la communauté ou aux représentations intentionnelles des données des communautés alliées. Les correspondances entre les modèles internes des pairs et le modèle d'un super-pair sont déduites en évaluant les mesures de similarité entre les éléments de ces modèles internes et ceux du modèle interne du super-pair. La matrice de correspondance SP/P pour un super-pair donné contient en colonne les nœuds du $sGraph$ du domaine et en ligne ceux des $sGraph$ des pairs.

	n_1^S	n_2^S	$\ldots$	n_n^S
n_1^T				$\cong$
n_2^T	$\cong$			
$\ldots$				
n_m^T			$\cong$	

Table 3.3 – Matrice de correspondance Super-pair/pair.

Un élément $MatSPP(n_g, n_l)$ de cette matrice indique la façon dont les nœuds correspondants sont liés. Nous ne considérons que les correspondances sémantiques exactes et directes (équivalence $\cong$). Deux éléments sont considérés comme équivalents si leur similarité globale est supérieure à un seuil ϵ_{acc} donné. Dans ce cas, une correspondance sémantique est un quadruplet :

$$< ID_{ij}, n_i^S, n_j^T, \cong \gamma_{ij} > \quad 1 \leq i \leq |S| \wedge 1 \leq j \leq |T|. \tag{3.17}$$

avec ID_{ij} identifiant unique de la correspondance, n_i^T et n_j^T respectivement les i-ème et j-ème nœuds des $sGraph$ concernés S et T, $|S|$ et $|T|$ les nombres de nœuds S et T, γ_{ij} valeur de la similarité. Dans le cas où il n'y a pas de correspondance, la relation entre n_i^S et n_j^T est évaluée à $null$. L'ensemble des correspondances sémantiques peut être vu comme un morphisme entre deux $sGraph$. Par conséquent, un tel morphisme est une relation binaire entre deux ensembles de nœuds, c'est-à-dire un ensemble de couples (n^S, n^T) tirés de $N_S \times N_T$. Le morphisme ne véhicule pas de sémantique par rapport à la transformation des instances des schémas.

Notre algorithme peut être vu comme un opérateur qui prend en entrée deux schémas bien formés et conformes au modèle $sGraph$ et produit en sortie un morphisme indiquant les correspondances sémantiques entre les nœuds des deux graphes. Il est basé sur les techniques linguistiques et structurelles introduites dans les sections précédentes pour l'évaluation de l'affinité entre nœuds en considérant leur noms et leurs contextes d'apparition dans leurs graphes respectifs.

Il peut être décomposé comme suit :

1. Détermination des étiquettes des nœuds et élimination des ambiguïtés linguistiques. Cette phase permet de déterminer le concept et les synonymes associés à chaque nœud. Cette phase inclut un processus de normalisation des étiquettes car des éléments similaires peuvent avoir des noms différents souvent à cause de la présence de ponctuations, d'abréviations, etc. Dans cette partie, les

noms sont transformés en entités atomiques sur la base des ponctuation, les majuscules, les prépositions et les articles éliminés, etc. Cette étape se base sur l'utilisation d'un thésaurus spécifique au domaine.

2. Détermination des relations sémantiques entre les nœuds et construction des vecteurs de voisinage sémantique pour chaque nœud. Seuls les nœuds dans le voisinage immédiat sont considérés.
3. Détermination des similarités linguistique et de voisinage des nœuds et enfin évaluation de la similarité globale.

L'algorithme ci dessous décrit le principe de construction de la matrice de correspondance SP/P. Pour chaque couple de nœuds des $sGraph$, les mesures de similarités sont évaluées et la correspondance initialisée par défaut à $null$ (1-8). Ensuite (9-16) pour chaque nœud n_l du $sGraph$ sGP du pair, on cherche l'élément $ngmax$ qui lui est le plus similaire dans le $sGraph$ $sGSP$ du super-pair. Si cette similarité est supérieure au seuil admis ϵ_{acc}, alors on a trouvé une équivalence $n_l \cong ngmax$.

Algorithme 1 Génération de la Matrice de correspondance Super-pair /Pair

Entrée : $sGSP$ et sGP $sGraph$ du super-pair et du pair.
ω_s poids des synonymes, ω_t poids des types et α poids de la linguistique
Sortie : $MatSPP$: Matrice de correspondance des deux $sGraph$.

1: **Pour tout** $n_g \in NoeudInterne(sGSP)$ **Faire**
2: **Pour tout** $n_l \in NoeudInterne(sGP)$ **Faire**
3: $x \leftarrow \omega_s \times sim_s(n_g, n_l) + \omega_t \times sim_t(n_g, n_l)$
4: $y \leftarrow sim_v(n_g, n_l)$
5: $sim(n_g, n_l) \leftarrow \alpha \times x + (1 - \alpha) \times y$
6: $add(MatSPP, n_g, n_l, null)$
7: **Fin Pour**
8: **Fin Pour**
9: **Pour tout** $n_l \in NoeudInterne(sGP)$ **Faire**
10: $ngmax \leftarrow n \in sGSP \,|\, \forall\, n_g \in\ sGSP, n_g \neq n\, sim(n_l, n) > sim(n_l, n_g)$
11: **Si** $sim(n_l, ngmax) > \epsilon_{acc}$ **Alors**
12: $add(MatSPP, ngmax, n_l, \cong)$
13: **Fin Si**
14: **Fin Pour**

3.6.5 Illustration du processus d'appariement

Dans cette partie, nous illustrons le processus de découverte des correspondances sémantiques, mais aussi comment ces dernières sont utilisées pour reformuler une requête d'un pair vers un autre. Considérons le scénario de la figure 3.12 dans lequel le système de partage de données du fleuve concerne les domaines *Agriculture* et *Pédologie*.

La connaissance de chaque domaine est décrite par un $sGraph$ dans lequel nous avons omis, pour des raisons de lisibilité, les étiquettes des arcs et les nœuds externes représentant les types. La connaissance du domaine *Agriculture* concerne les sols et les cultures pratiquées sur ces sols, tandis que celle du domaine *Pédologie* concerne les sols. De plus, le domaine *Agriculture* contient deux pairs, SAED (Société d'Aménagement et d'Exploitation des terres du Delta) et ISRA (Institut Sénégalais de Recherche Agronomique) publiant aussi leurs données sous forme de $sGraph$. Nous n'avons pas matérialisé les pairs du domaine *Pédologie* par souci de lisibilité. Les deux domaines ne sont pas disjoints puisque aussi

Algorithme 2 Génération de la Matrice de correspondance Super-Pair/Super-Pair

Entrée : $sGSP1$ et $sGSP2$ $sGraph$ respectif de $SP1$ et de $SP2$.
Sortie : $MatSPSP$: Matrice de correspondance des deux $sGraph$
Pour tout $n_g^1 \in NoeudInterne(sGSP1)$ **Faire**
  Pour tout $n_g^2 \in NoeudInterne(sGSP2)$ **Faire**
    $x \leftarrow \omega_s \times sim_s(n_g^1, n_g^2) + \omega_t sim_t \times sim_t(n_g^1, n_g^2)$
    $y \leftarrow sim_v(n_g^1, n_g^2)$
    $sim(n_g^1, n_g^2) \leftarrow \alpha \times x + (1 - \alpha) \times y$
    Si $sim(n_g^1, n_g^2) > \epsilon_{acc}$ **Alors**
      $add(MatSPP, n_g^1, n_g^2, \cong)$
    Sinon
      $add(MatSPP, n_g^1, n_g^2, null)$
    Fin Si
  Fin Pour
Fin Pour

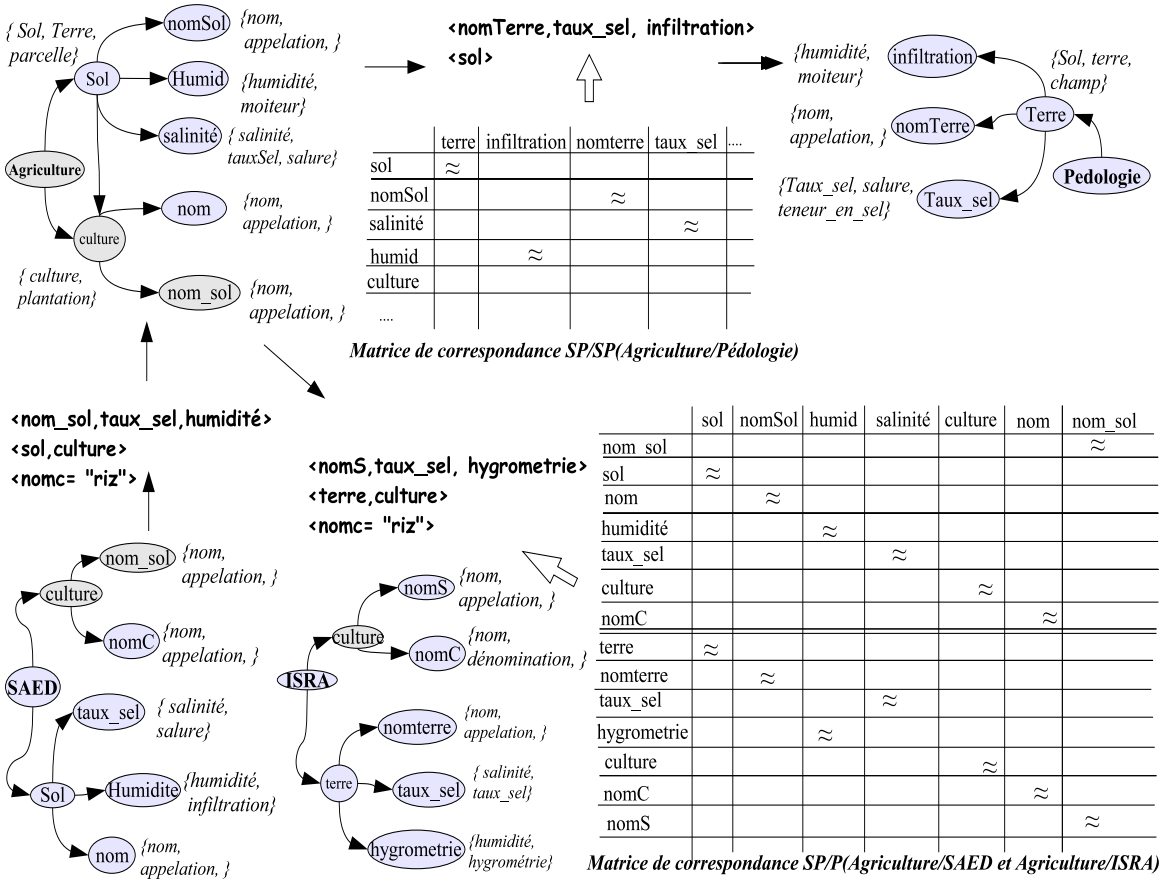


Figure 3.12 – Médiation de données des domaines Agriculture et Pédologie. ISRA(Institut Sénégalais de Recherche Agronomique), SAED (Société d'Aménagement et d'Exploitation du Delta).

bien les experts de l'agriculture que ceux de la pédologie manipulent des données concernant les sols. Les correspondances entre ces domaines sont alors stockées dans la matrice de correspondance SP/SP

(Agriculture/Pédologie). Par ailleurs, les pairs SAED et ISRA décrivent des données sur les sols et les cultures mais avec des vocabulaires différents de celui suggéré pour leur communauté par le super-pair Agriculture. Les correspondances déduites sont stockées dans la matrice de correspondance SP/P (Agriculture/SAED et Agriculture/ISRA). Toutes ces correspondances sont déduites grâce aux mesures de similarité présentées dans la partie précédente.

Par exemple, pour établir la correspondance $< taux\_sel, salinite, \cong >$ de la matrice de correspondance SP/SP (Agriculture/Pédologie), nous calculons la similarité $Sim(taux\_sel, salinite)$ entre les nœuds $taux\_sel$ et $salinite$ (pour $\lambda_l = \lambda_v = \omega_s = \omega_t = 0.5$) comme suit :

$$Sim(taux\_sel, salinite) = 0.5 \times S_l(taux\_sel, salinite) + 0.5 \times S_v(taux\_sel, salinite)$$

avec

$$S_l(taux\_sel, salinite) = 0.5 \times S_s(taux\_sel, salinite) + 0.5 \times S_t(taux\_sel, salinite).$$

En considérant que $taux\_sel$ et $salinite$ sont de même type, la similarité de type est alors maximale et vaut 1. En prenant par exemple $\alpha = 0.5$ dans l'équation (3.13), la similarité de synonymes s'exprime comme suit :

$$S_s(taux\_sel, salinite) = \frac{|\{tauxsel, salure\}|}{|\{tauxsel, salure\}| + 0.5|\{teneur\_en\_sel\}| + 0.5|\{salinite\}|} = \frac{2}{3}.$$

Par conséquent

$$S_l(taux\_sel, salinite) = 0.5 \times \frac{2}{3} + 0.5 \times 1 = 0.83.$$

La similarité de voisinage $Sim_v(taux\_sel, salinite)$ s'exprime ainsi :

$$S_v(taux\_sel, salinite) = \frac{|\{x \in V(taux\_sel)/\exists y \in V(salinite) \wedge Sim(x, y) > \epsilon_{acc}\}|}{|V(taux\_sel) \cup V(salinite)|}.$$

Nous avons $V(salinite) = \{sol\}$, $V(taux\_sel) = \{terre\}$, $syn(sol) = \{sol, terre, parcelle\}$ et $syn(terre) = \{sol, terre, champ\}$. Nous montrons que $S_s(sol, terre) = 0.5$ et $S_t(sol, terre) = 1$. Nous trouvons alors :

$$S_l(sol, terre) = 0.5 \times 0.5 + 0.5 \times 1 = 0.75 > \epsilon_{acc} = 0.6$$

Par substitution $S_v = \frac{1}{2}$. La similarité globale vaut alors :

$$Sim(taux\_sel, salinite) = 0.5 \times 0.83 + 0.5 \times 0.5 = 0.67$$

Supposons maintenant que le pair SAED exprime la requête suivante :

Quels sont le nom, le taux de salinité et l'humidité des sols sur lesquels le riz est cultivé?

Le pair SAED ne connaissant que son schéma, la requête est exprimée sur son propre schéma avec son propre vocabulaire :

(R1)

$< nom\_sol, taux\_sel, humidite >$

$< sol, culture >$

$< nomc = "riz" >$

La requête (R1) est ensuite envoyée au super-pair du domaine *Agriculture*. Dans un premier temps, elle est réécrite au sein du domaine. Ainsi, d'après la matrice de correspondance SP/P (Agriculture/SAED et Agriculture/ISRA), la requête est réécrite avec le vocabulaire du pair ISRA comme suit :
(R2)
$< nomS, taux\_sel, hygrometrie >$
$< terre, culture >$
$< nomc = "riz" >$.

La requête (R2) peut maintenant être reformulée avec le langage d'interrogation du pair ISRA et les résultats envoyés au super-pair *Agriculture*.

D'autre part, la matrice de correspondance SP/SP (Agriculture/Pédologie) permet de réécrire la requête (R1), avec le vocabulaire du domaine *Pédologie*, en une requête (R3):
(R3)
$< nomTerre, taux\_sel, infiltration >$
$< sol >$

Celle-ci est évaluée dans le domaine et les résultats sont ensuite envoyés au Super-pair *Agriculture* qui peut enfin composer les résultats et les envoyer au pair SAED ayant initié la requête.

3.7 Bases de la topologie sémantique

Dans cette partie, nous détaillons le processus de formation de la topologie sémantique au dessus du réseau physique existant. La topologie est formée durant la période de découverte des correspondances sémantiques. Cette topologie sémantique servira de base à la propagation des requêtes. En effet, le passage à l'échelle d'un système P2P est essentiellement déterminé par la façon dont les requêtes sont propagées à travers le réseau. Comme nous l'avons mentionné dans l'état de l'art, il est d'un commun accord que les systèmes P2P qui diffusent les requêtes à tous les pairs ne passent pas à l'échelle. Ainsi donc, un mécanisme de routage de requête intelligent est nécessaire pour n'envoyer les requêtes qu'aux pairs susceptibles d'y répondre. La topologie sémantique permet de matérialiser des chemins sémantiques entre pairs, chemins qui pourront être utilisés plus tard pour le routage des requêtes. Cette topologie est induite par les correspondances sémantiques mais aussi par la notion d'expertise de (super-)pair que nous allons introduire dans ce qui suit.

3.7.1 Expertise

A la jonction de son super-pair, un pair enregistre, en plus de son schéma et de ses autres caractéristiques, une description des éléments qu'il souhaite partager. Cette description, appelée *expertise*, constitue un résumé du $sGraph$ du pair en termes de couples de nœuds supportés. Ce résumé du schéma sera un moyen rapide pour le super-pair d'évaluer la capacité d'un pair donné à prendre en charge une requête plutôt que d'avoir à comparer le graphe de la requête avec le $sgraph$ du pair, ce qui est potentiellement coûteux en temps.

Définition 3.9. [***Expertise.***] L'expertise d'un pair P renseigne sur la capacité d'un pair à fournir certaines données. Elle est définie à partir du $sGraph$ SG du pair concerné, comme suit :

$$Exp(P) = \{(n_p, m_p) \in SG \mid Contains(n_p, m_p) \vee Associates(n_p, m_p) \vee isA(n_p, m_p)\}. \quad (3.18)$$

Par exemple, l'expertise du pair P de la Figure 3.7 peut être exprimée ainsi :

```
{(sol,nom_sol),(sol,salinité),(sol,humidité),(sol,culture),
(culture,nom),(culture,rendement),(culture,sol),(culture,cycle)}.
```

Ces expertises sont stockées par le super-pair dans deux types de tables : $E_{SP/SP}$ pour les expertises des super-pairs *parrains* des communautés liées et $E_{SP/P}$ pour les expertises des pairs de la communauté dont le super-pair est le *parrain*. Les tables d'expertise sont mises à jour régulièrement en tenant en compte l'aspect dynamique du réseau qui résulte des départs et de l'arrivée des pairs. Ainsi donc, chaque (super-)pair informe le réseau P2P de la description sémantique de son expertise. Par conséquent, la connaissance à propos des expertises et les matrices de correspondance forment une topologie sémantique indépendante du réseau physique. Les expertises sont utilisées plus tard pour aider à la sélection des pairs pertinents en faisant une mise en correspondance entre expertises et sujets (résumés) de requêtes. Étant donné que chaque super-pair collecte les expertises des pairs qu'il parraine, chaque pair connaît implicitement toutes les expertises qui sont localisées dans la même communauté sémantique.

3.7.2 Création d'une communauté

Une communauté sémantique apparaît quand un nouveau super-pair SP_j appelé *parrain de communauté* joint le PDMS en suggérant un schéma SS_j. Le super-pair définit ensuite son expertise et publie une annonce pour informer de sa venue tout en attendant les manifestations d'intérêts des super-pairs liés et les demandes d'adhésion à la communauté de certains pairs. L'annonce d'une communauté sémantique est faite par son super-pair *parrain* SP_j qui envoie une annonce $SCA_j = (CID, SS_j, E_j, T_j, \epsilon_{acc}, TTL)$ à ses voisins. Ce message est un fichier XML contenant l'ID CID de la communauté, le schéma suggéré SS_j, l'expertise correspondante E_j, le thème d'intérêt de la communauté T_j, la valeur minimum de similarité ϵ_{acc} exigée pour établir une correspondance sémantique entre le schéma suggéré par le parrain et les schémas des autres (super-)pairs. Pour optimiser la communication et l'envoi de messages durant la fondation des communautés, nous combinons un algorithme de diffusion sous contrainte [113], qui diminue le nombre de messages en doublons, avec un TTL qui réduit le rayon de propagation de l'annonce. A la réception d'une annonce de communauté, un super-pair SP_r déclenche le processus médiation sémantique pour trouver les correspondances entre le schéma qu'il suggère et le schéma suggéré reçu SS_j. Les correspondances sémantiques trouvées sont stockées dans la matrice de correspondance $\mathcal{M}^{j,r}_{SP/SP}$ de SP_r et envoyées aussi à SP_j qui peut les approuver ou les rejeter. En cas d'acceptation, chaque super-pair ajoute l'expertise de l'autre dans sa table d'expertise $E_{SP/SP}$. SP_r fait ensuite suivre l'annonce SCA_j aux pairs qui l'avait auparavant informé de leur intérêt pour le thème T_j. Par la suite, la confiance accordée à la relation sémantique entre les super-pairs SP_j et SP_r est évaluée grâce à une mesure de *confiance* de la matrice de correspondance. Cette mesure de confiance est composée et elle est calculée en utilisant les mesures de similarités des éléments participant à la matrice de correspondances comme suit :

$$Conf(\mathcal{M}^{j,r}_{SP/SP}) = \frac{1}{|\mathcal{M}^{j,r}_{SP/SP}|} \sum_{\mathcal{M}^{j,r}_{SP/SP}(n^{SS_j}, m^{SS_r})} Sim(n^{SS_j}, m^{SS_r}). \tag{3.19}$$

La fonction $Conf(\mathcal{M}^{j,r}_{SP/SP})$ permet à un super-pair de dresser la liste de ses *k-meilleurs* alliés. Cette liste qui, par la suite, est propagée à l'ensemble des pairs de la communauté, constitue un ensemble de

suppléants vers lesquels les pairs pourront se tourner en cas d'indisponibilité du super-pair et ceci pour ne pas être isolés du reste du PDMS.

3.7.3 Adhésion à une communauté

Chaque pair P_i pair sollicitant le partage de données envoie une demande d'adhésion $MA_i = (PID, S_i, E_i, T)$ aux communautés sémantiques déjà mises en place, par le biais de leurs super-pairs parrains. Ce message contient l'ID PID du pair, son thème d'intérêt T, son $sGraph$ S_i, et son expertise E_i et d'autres informations telles que sa bande passante, son adresse IP, etc. Si le parrain de P_i n'a pas encore joint le réseau, l'annonce MA_i devra être mémorisée par le point d'accès de telle sorte que ce dernier puisse informer le pair *orphelin* dès que son *parrain* entre dans le PDMS.

Par contre, si le *parrain* de sa communauté sémantique a été trouvé et si c'est la première fois que le pair joint la communauté, le processus de médiation sémantique entre le $sGraph$ du pair et le schéma suggéré par le parrain est entamé. En cas d'acceptation de la demande d'adhésion , le super-pair *parrain* stocke les correspondances trouvées dans sa matrice de correspondance $M_{SP/P}$ et l'expertise du pair dans sa table d'expertise $E_{SP/P}$. Le parrain envoie ensuite une notification d'acceptation à P_i. Les autres caractéristiques telles que l'IP, la bande passante, etc., sont quant à elles, mémorisées dans l'index du super-pair. L'inscription est valide pour un intervalle de temps dénoté TTL (*Time-To-Live*). Chaque pair doit rappeler au super-pair qu'il est *encore en vie* avant que le TTL n'expire s'il tient à ce que ses données soient encore disponibles à l'interrogation pour le reste du réseau. Le TTL est défini par le pair et doit figurer dans un interval fixé par le super-pair. Il indique la fréquence à laquelle le super-pair souhaiterait avoir des informations sur la survie de ses pairs. En cas d'expiration du TTL et d'absence de signe de vie du pair ou en cas de départ du pair du réseau, toutes ses références sont supprimées du catalogue, de la table d'expertises et de la matrice de correspondance du super-pair.

Si par contre le pair se réinscrit au niveau du super-pair parce qu'il veut informer de sa survie, ce dernier vérifie d'abord si son schéma a été modifié (ceci peut être indiqué par le pair en envoyant un drapeau spécifiant si son schéma a été modifié ou pas). Dans le cas échéant, le super-pair découvre la partie de la nouvelle l'expertise qui ne figure pas dans la table d'expertise $E_{SP/P}$ actuelle et déclenche le processus de découverte de correspondances pour ces éléments et définit un nouveau TTL. Dans le cas contraire, seul le TTL est redéfini.

Comme pour le cas inter-communauté, la *confiance* en l'appariement du schéma du pair avec celui de sa communauté est estimée avec une fonction de confiance tenant en compte les valeurs des mesures de similarité entre les nœuds supportés par le pair et ceux du schéma suggéré de la communauté.

$$Conf(\mathcal{M}^{i,k}_{SP/P}) = \frac{1}{|\mathcal{M}^{i,k}_{SP/P}|} \sum_{\mathcal{M}^{i,k}_{SP/P}(n^{S_i}, m^{SS_k})} Sim(n^{S_i}, m^{SS_k}). \tag{3.20}$$

La valeur de la *confiance* est valable tant que le TTL indiquant la fraîcheur des données n'a pas expiré. En invalidant les inscriptions des pairs périodiquement, nous obtenons un comportement dynamique similaire aux protocoles pour des configurations dynamiques (par exemple, DHCP) dans lesquels les nœuds peuvent quitter sans notification.

3.7.4 Graphe d'accointances

Le processus de découverte de correspondances sémantiques, combinée aux tables d'expertises, induit une topologie sémantique au dessus du réseau physique. Cette topologie sémantique est appelée

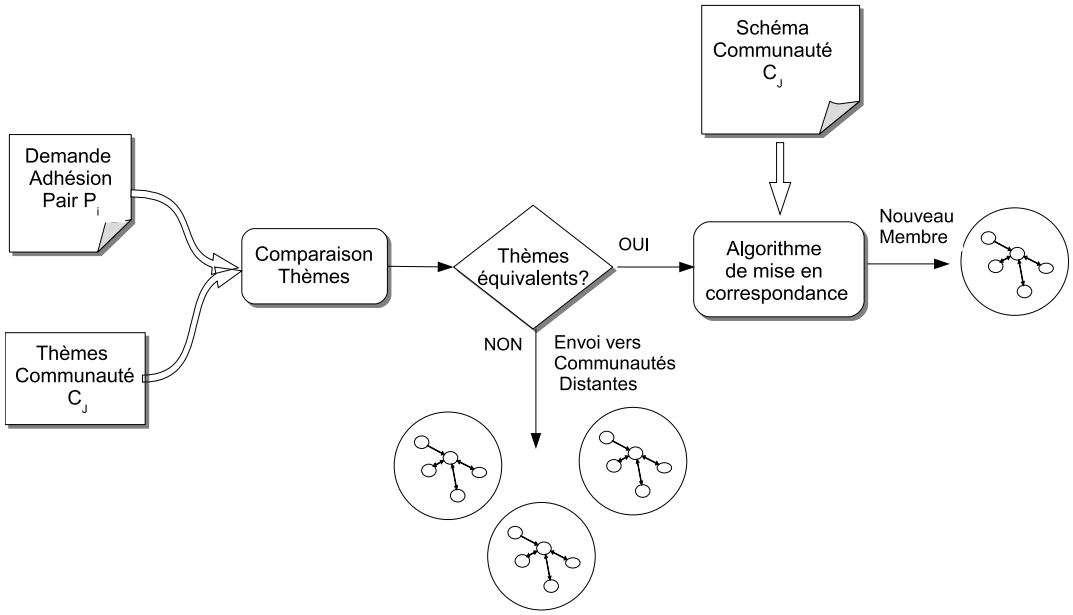


Figure 3.13 – Adhésion à une communauté.

graphe d'accointances car représentant les affinités sémantiques inter-communautés et les appartenances des pairs à des communautés spécifiques.

Définition 3.10. [**Graphe d'accointances sémantiques**]

Soit $\mathcal{P} \cup \mathcal{SP}$ un ensemble de pairs et de super-pairs, un graphe d'accointances sémantique sur $\mathcal{P} \cup \mathcal{SP}$ est un graphe $\Gamma = (\mathcal{P} \cup \mathcal{SP}, \mathcal{A})$ avec $\mathcal{P} \cup \mathcal{SP}$ ensemble de sommets et $\mathcal{A} \subseteq (\mathcal{P} \cup \mathcal{SP}) \times (\mathcal{P} \cup \mathcal{SP})$ un ensemble d'arcs étiquetés tel que pour tout arc $(P, R) \in \mathcal{A}$, les (super-)pairs P et R approuvent la qualité $Conf$ de l'appariement de leurs schémas. Un arc (P, SP) exprime l'appartenance du pair P à la communauté parrainée par le super-pair SP tandis que arc (SP_i, SP_j) ($i \neq j$) indique l'alliance entre les communautés dirigées par les super-pairs SP_i et SP_j.

Un graphe d'accointances avec n sommets peut être représenté par une matrice d'adjacence $B = (b_{ij})$ avec des valeurs égales au poids des arcs et indiquant les forces des relations sémantiques entre les pairs représentés par les sommets liés par les arcs :

$$b_{ij} = \begin{cases} Conf(H_i, H_j) \text{ avec } (H_i, H_j) \in (\mathcal{P} \cup \mathcal{SP})^2 & \text{Si } (H_i, H_j) \in \mathcal{A} \\ 0 & \text{Si } (H_i, H_j) \notin \mathcal{A} \end{cases} \tag{3.21}$$

avec $1 \leq i \leq |\mathcal{P} \cup \mathcal{SP}|$ et $1 \leq j \leq |\mathcal{P} \cup \mathcal{SP}|$.

3.7.5 Exemple

La figure 3.14 illustre le rôle de la médiation sémantique dans la formation des communautés sémantiques dans une partie de PDMS avec les communautés sémantiques suivantes : *Agriculture*, *Pédologie*, *Élevage* et *Santé*. Par souci de simplification, nous avons donné une représentation minimale de chaque

$sGraph$. Supposons que SP_A joint le PDMS pour définir la communauté *Agriculture*. Il envoie une annonce de communauté SCA_A à ses voisins logiques (c'est-à-dire, SP_E et SP_S). En accord avec l'algorithme de diffusion sous contrainte et un TTL, l'annonce est propagée à SP_P et aux autres pairs dont SP_E et SP_S se rappellent avoir reçu leurs annonces d'intérêt pour la nouvelle communauté, ici les pairs P_{SAED} et P_{ISRA}. Super-Pair SP_P répond à SP_A (parce que les thèmes *Agriculture* and *Pédologie* sont liés) en envoyant une manifestation d'intérêt IR_P contenant les correspondances trouvées durant le processus de médiation sémantique, avant d'attendre une confirmation de l'acceptation. Ensuite, les correspondances trouvées sont ajoutées dans la matrice de correspondance $\mathcal{M}_{SP/SP}$ des super-pairs concernés. SP_E and SP_S ne sont pas intéressés par l'invitation et donc ne répondent pas à SP_A. P_{SAED} et P_{ISRA} intéressés par la communauté *Agriculture* envoient leurs demandes d'adhésion MA_1 et MA_2 à SP_A, qui va déclencher le processus de découverte de correspondances entre les schémas des pairs et le schéma qu'il suggère. SP_A envoie une confirmation d'adhésion à P_{SAED} et P_{ISRA} en cas de réussite de la mise en correspondance et ajoute les correspondances trouvées dans sa matrice de correspondance $M_{SP/P}$.

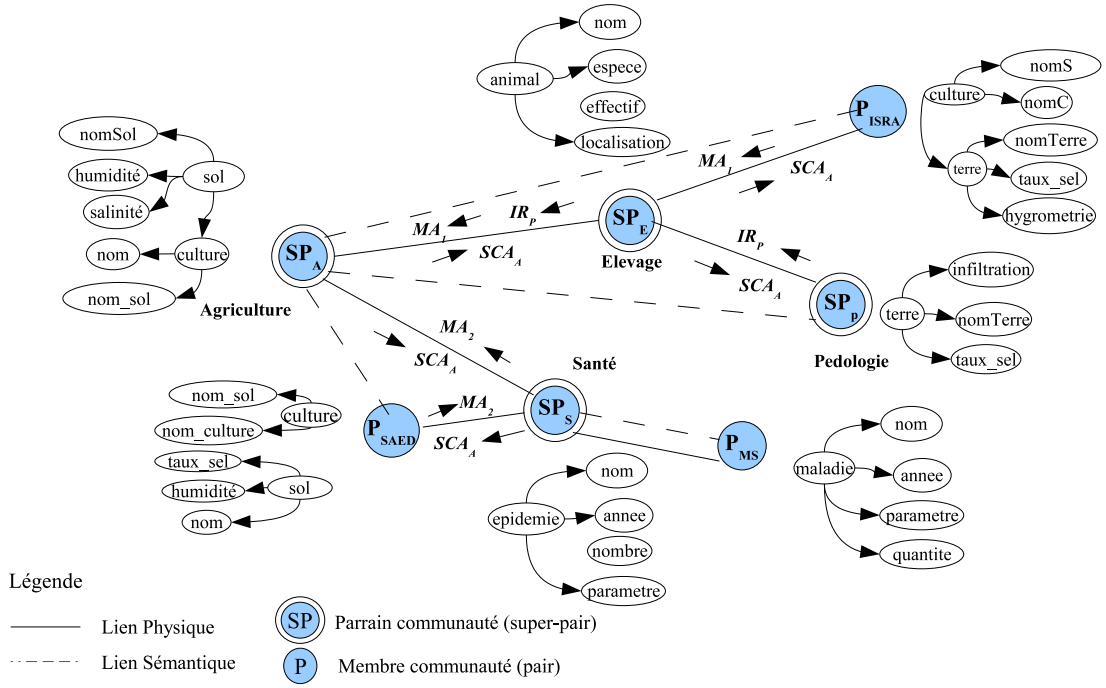


Figure 3.14 – Fondation des communautés sémantiques durant le processus de découverte de correspondances.

3.7.6 Détection et gestion des fautes

Dans notre modèle de communication, le super-pair constitue le point d'accès de ses pairs au réseau. Avec cette considération, dès que le super-pair devient indisponible ou quitte le réseau, tous les pairs qui lui sont rattachés seront isolés. Pour augmenter la robustesse, nous introduisons une auto-organisation pour faire face à une situation où les super-pairs seraient indisponibles. Nous nous basons sur le fait que tous les nœuds, à la fois les pairs et les super-pairs ont des identificateurs uniques permettant de les localiser. Nous utilisons en plus de cela le fait que les pairs et les super-pairs s'envoient des messages

pour montrer qu'ils sont toujours en ligne. Nous supposons aussi que les super-pairs envoient à leurs pairs la liste de leurs *k-meilleurs alliés* de sorte que ces pairs puissent se tourner vers ces derniers au cas où ils ne seraient plus disponibles.

Par conséquent, si un super-pair devient indisponible, tous ses pairs entrent provisoirement dans une *communauté fictive* regroupant tous les pairs isolés par le fait de l'indisponibilité de leur parrain. Chacun de ces pairs peut alors contacter l'un des *k-meilleurs alliés* connus par son parrain pour espérer être indexé. Le parrain contacté peut accepter ou refuser la connection suivant sa charge actuelle, qui est déterminée par sa capacité de calcul, de stockage, mais aussi sa bande passante.

Les fautes concernant les pairs sont détectés de la même façon que ceux concernant les super-pairs. Si un pair devient inaccessible ou quitte subitement le réseau, son parrain super-pair supprime les entrées de son index le concernant. Si le pair quitte normalement le réseau, il avise juste son super-pair, qui va se charger de supprimer toutes les entrées le concernant dans son index. Si par contre le pair tombe en panne du fait d'une faute inattendue, ceci est détecté par le super-pair quand ce dernier tarde à recevoir le message de ré-enregistrement du pair après l'expiration du TTL indiquant la fraîcheur des données.

3.7.7 Sémantique de l'intégration de données

Dans ce paragraphe, nous introduisons la sémantique du partage de données en accord avec la sémantique des sources de données des pairs, les matrices de correspondance et les schémas suggérés pour les communautés sémantiques.

Soit Γ un ensemble de constantes partagées par toutes les sources de données. Chaque source de données de pair se conformant à un modèle de données spécifique peut être vue comme une instance de $sGraph$. Soit $S = \{\mathcal{G} = (N_S, E_S), \xi, \Re, \eta, \ell\}$ un $sGraph$, pour des raisons de commodité, nous la dénoterons par l'expression $S = (N_S, E_S, \eta_S, \ell_S)$.

Définition 3.11. [**Graphe d'instance de** $sGraph$]. Soit $S = (N_S, E_S, \eta_S, \ell_S)$ un $sGraph$, une instance de S est un graphe $I(S) = (N_I, E_I, \eta_I, \ell_I)$ où N_I est un ensemble de nœuds, E_I un ensemble d'arcs et :

(1) η_I est une fonction d'étiquetage $\eta_I : N_I \rightarrow \Gamma$ telle que
$\forall n \in N_I, \eta_I(n) \in \Gamma \cup \{\epsilon\}$,

(2) ℓ_I est une fonction d'étiquetage $\ell_I : E_I \rightarrow E_S$ telle que
$\forall (n_i, n_j) \in E_I, \ell_I(n_i, n_j) = \ell_S(\delta(n_i), \delta(n_j))$,

(3) δ est une fonction de typage $\delta : N_I \rightarrow N_S$, telle que
$\forall n \in N_I, \delta(n) \in N_S$, et $\forall (n_i, n_j) \in E_I, (\delta(n_i), \delta(n_j)) \in E_S$.

Nous définissons la sémantique des correspondances dans le cadre du processus de découverte de correspondances intra et inter communautés.

Supposons un PDMS $\mathfrak{P} = (\mathcal{G}, \mathcal{M})$ avec un ensemble de matrices de correspondance $\mathcal{M}$. Soit C une communauté sémantique avec un schéma suggéré SS, un ensemble de pairs affiliés d'ensemble de schémas $S_1, \ldots, S_n$ (respectivement associés aux sources de données $I(S_1), \ldots, I(S_n)$), un ensemble de super-pairs alliés d'ensemble de schémas suggérés $SS_1, \ldots, SS_m$ et $\{\mathcal{M}_{SP/SP}, \mathcal{M}_{SP/P}\}$ l'ensemble de matrices de correspondance associées. Nous pouvons définir la sémantique des correspondances à travers le concept de valuation (locale et distante).

Définition 3.12. [**valuation locale**] Soit $r_i = (n_i^C, n_{i_1}, \ldots, n_{i_n})$ une ligne de la matrice de correspondance intra-communauté $\mathcal{M}_{SP/P}$. Étant donné une communauté sémantique avec un schéma suggéré SS et d'ensemble de pairs affiliés d'ensemble de schémas $S_1, \ldots, S_n$ respectivement associés aux sources de données $I(S_1), \ldots, I(S_n)$ une valuation locale de r_i dans la communauté sémantique est une fonction

φ qui fait correspondre r_i à un a tuple $(n_i^C, \vartheta_{i_1}, \ldots, \vartheta_{i_n})$ où $n_i^C \in SS$ et $\vartheta_{i_j} \in I(S_j)(1 \leq j \leq n)$, telle que $\delta_{I(S_j)}(\vartheta_{i_j}) = n_{i_j}(1 \leq j \leq n)$. La sémantique de la matrice de correspondance $\mathcal{M}_{SP/P}$ (avec p lignes) est donnée par la conjonction des valuations locales au sein de la communauté comme suit :

$$\bigwedge_{i=1}^{p} \Big(\varphi(n_i^C) = \varphi(n_{i_1}) \cup \ldots \cup \varphi(n_{i_n})\Big) \quad \text{avec} \quad n_{i_k} \in S_k \quad \text{telle que}$$
$$(n_{k_j}, n_{k_l}) \in E_{S_k} \Leftrightarrow (\varphi(n_{k_j}), \varphi(n_{k_l})) \in E_{I(S_k)} \forall j \in [1, .., n] \tag{3.22}$$

Définition 3.13. [**valuation distante**] Soit $t_j = (n_j^C, n_{j_1}^{C_{j_1}}, \ldots, n_{j_m}^{C_{j_m}})$ une ligne de la matrice de correspondances inter-communauté $\mathcal{M}_{SP/SP}$, avec $n_j^C \in SS$ et $n_{j_k}^{C_{j_k}} \in SS_k$. Étant donné une communauté sémantique de schéma suggéré SS et d'ensemble de super-pairs alliés de schémas suggérés $SS_1, \ldots, SS_n$, une valuation distante de t_i depuis les communautés alliées est une fonction φ qui fait correspondre t_j à un tuple $(n_j^C, \vartheta_{j_1}, \ldots, \vartheta_{j_n})$ où $n_j^C \in SS$ et ϑ_{j_k} est une valuation locale du concept $n_{j_k}^{\mathcal{C}_{j_k}}$ qui correspond à n_j^C dans la communauté sémantique C_{j_k}. La sémantique de la matrice de correspondance $\mathcal{M}_{SP/SP}$ (avec q lignes) est donnée par la conjonction des valuations distantes venant des communautés alliées comme suit :

$$\bigwedge_{j=1}^{q} \Big(\varphi(n_j^C) = \varphi(n_{j_1}^{C_{j_1}}) \cup \ldots \cup \varphi(n_{j_m}^{C_{j_m}})\Big) \quad \text{avec} \quad n_{j_k}^{\mathcal{C}_{j_k}} \in \mathcal{SS}_k. \tag{3.23}$$

3.8 Conclusion

Dans ce chapitre, nous avons introduit l'architecture de SenPeer et le processus de médiation de données. Le système est basé sur une topologie super-pair avec une organisation des pairs en communautés sémantiques. Chaque pair dispose de son propre schéma conforme à un modèle de données quelconque mais aussi de son propre langage d'interrogation. Pour venir à bout de l'hétérogénéité sémantique et syntaxique, les schémas sont exportés dans un format commun et les correspondances sémantiques établies au sein d'une même communauté, mais aussi avec les communautés alliées. Ces correspondances sont déduites grâce à un certain nombre mesures de similarité sémantique combinant les connaissances, aussi bien linguistiques que structurelles, sur les éléments des schémas. Nous avons aussi introduit le concept d'expertise[35] de pair représentant un résumé sommaire du schéma du pair et destiné à guider la sélection des pairs pertinents. L'ensemble des expertises des (super-)pairs, en combinaison avec les matrices de correspondance sémantique est à la base d'une topologie sémantique au dessus du réseau physique et dans laquelle les (super-)pairs ayant des schémas similaires forment un voisinage sémantique. Cette topologie est à la base du mécanisme de routage sémantique présenté dans le chapitre 4.

Certes, le passage à l'échelle est très important pour les PDMS mais nous ne pouvons pas obtenir un PDMS qui passe à l'échelle d'un système de partage de fichiers, étant donné que dans la réalité nous ne pouvons pas partager autant de sources de données que dans un système P2P de partage de fichiers. Nous avons choisi d'autoriser des schémas hétérogènes et plusieurs modèles de données, pour nous rapprocher plus d'un scénario réel. Par conséquent, la tâche devient plus difficile que dans le cas où le schéma est homogène et le modèle de données unique. D'ailleurs, ce dernier cas a tendance à plus passer à l'échelle, étant donné qu'il n'y a pas réellement d'effort de médiation sémantique. Cependant avec l'organisation en communautés sémantiques, nous évitons l'effort de découverte de correspondances quadratique en accord avec la taille du réseau. De plus, la découverte des correspondances n'est pas faite aveuglément

entre les schémas, mais elle est guidée par les communautés sémantiques. Les pairs joignant le réseau progressivement, l'intégration est faite à un premier niveau abstrait entre les thèmes, et à un niveau plus concret entre les schémas, ce qui permet de réduire le nombre de mises en correspondances et d'augmenter la scalabilité. Bien entendu, cette scalabilité dépend du nombre de pairs et de super-pairs, de la capacité des super-pairs mais aussi de la distribution des pairs entre communautés.

CHAPITRE 4

Traitement des requêtes

4.1 Introduction

Ce chapitre décrit le processus de traitement de requêtes dans SenPeer. Les requêtes sont initiées par les pairs sur leurs propres schémas et avec leurs propres langages d'interrogation. Elles peuvent être exécutées sur les schémas des autres pairs distants et avec les langages d'interrogation de ces derniers. En présence de plusieurs langages d'interrogation, les requêtes sont échangées dans un format commun d'échange de requêtes et ceci pour éviter les multiples traductions entre langages de requêtes. Afin de ne faire suivre une requête qu'au sous-ensemble de pairs susceptibles d'y répondre correctement, nous proposons un algorithme de routage sémantique s'appuyant sur la topologie sémantique induite par le processus de réconciliation sémantique. Dans ce cas, les requêtes sont reformulées du schéma source vers le schéma cible par un algorithme de reformulation de requêtes et ceci sur la base des correspondances sémantiques entre les éléments des schémas. En accord avec les informations sur les localisations des (super-)pairs pertinents pour la requête, les super-pairs génèrent des plans de requêtes dynamiques et distribués qui sont ensuite optimisés en tenant compte de leurs coûts de traitement et d'envoi, mais aussi de leurs temps de réponse. Le modèle de coût tient compte des statistiques sur les requêtes antérieures, des tailles résultats intermédiaires, mais aussi de la charge de travail dans les différents nœuds devant participer à la requête.

4.2 Processus d'interrogation

4.2.1 Idées de base du processus d'interrogation

Dans SenPeer, une requête est formulée par un pair sur son schéma et donc dans le langage d'interrogation de son schéma (SQL, XQuery, etc.). Les réponses attendues ne sont pas seulement ses données locales mais aussi celles des pairs de sa communauté ou des communautés alliées distantes. Dans le premier cas, l'interrogation se fait comme dans un système de gestion de données traditionnel. Dans le second cas, la requête est envoyée au parrain du pair qui va se charger de superviser son exécution au sein de la communauté, mais aussi de l'envoyer vers les communautés alliées pertinentes pour la requête. Nous pouvons résumer le traitement des requêtes par les étapes suivantes:

1. Une requête est exprimée par un utilisateur à l'aide de l'interface graphique utilisateur du pair, sur le schéma du pair mais aussi avec son langage d'interrogation. La requête est d'abord exécutée sur les données locales comme dans un système de gestion de données traditionnel. Si les résultats de cette phase ne sont pas suffisants (par exemple l'ensemble de réponses est vide), la phase d'interrogation du reste du réseau est initiée.

2. La requête est reformulée dans un format d'échange de requêtes appelé SQUEL(*SenPeer Query Exchange Language*). Elle est ensuite envoyée à son parrain super-pair. En accord avec les informations que le pair initiateur de la requête avait envoyé au super-pair en intégrant la communauté, le parrain extraie le sujet de la requête. Le sujet de la requête est un résumé de la requête exprimé dans le même formalisme logique que les expertises détenues par le super-pair.
3. En s'aidant de la topologie sémantique formée lors de la phase de réconciliation sémantique, le parrain procède à un routage sémantique de la requête en se basant sur les (super-)pairs pertinents pour la requête. La capacité des pairs à prendre en charge une requête est évaluée en comparant le sujet de la requête avec l'ensemble des expertises des voisins sémantiques stockées dans les tables d'expertises intra et inter-communautés. Cette étape produit une requête sémantique annotée avec l'ensemble des pairs ou super-pairs pouvant la prendre en charge.
4. Chaque sous requête est réécrite du vocabulaire du pair initiateur de la requête vers ceux des (super-)pairs devant contribuer à la réponse finale. Ceci est fait par le biais d'un algorithme de reformulation de requête tenant en compte les matrices de correspondance $\mathcal{M}_{SP/SP}$ et $\mathcal{M}_{SP/P}$.
5. La requête annotée est ensuite passée au générateur de plan qui a la charge de créer un plan de requête distribuée qui va superviser l'exécution de la requête distribuée en tenant compte de la distribution des (super-)pairs contribuant à la requête. Ce plan est ensuite optimisé en tenant compte des coûts de transmission des données entre les nœuds contribuant à la requête mais aussi le coût traitement de chaque opérateur. Les sous-plans locaux à la communauté sont ensuite instanciés et distribués aux pairs concernés tandis que ceux concernant les communautés distantes sont envoyés vers les parrains de ces communautés qui vont ensuite continuer le routage et l'optimisation au sein de leurs communautés respectives.
6. Les résultats de chacun de ces sous-plans sont ensuite fusionnés puis renvoyés au pair ayant initié la requête.

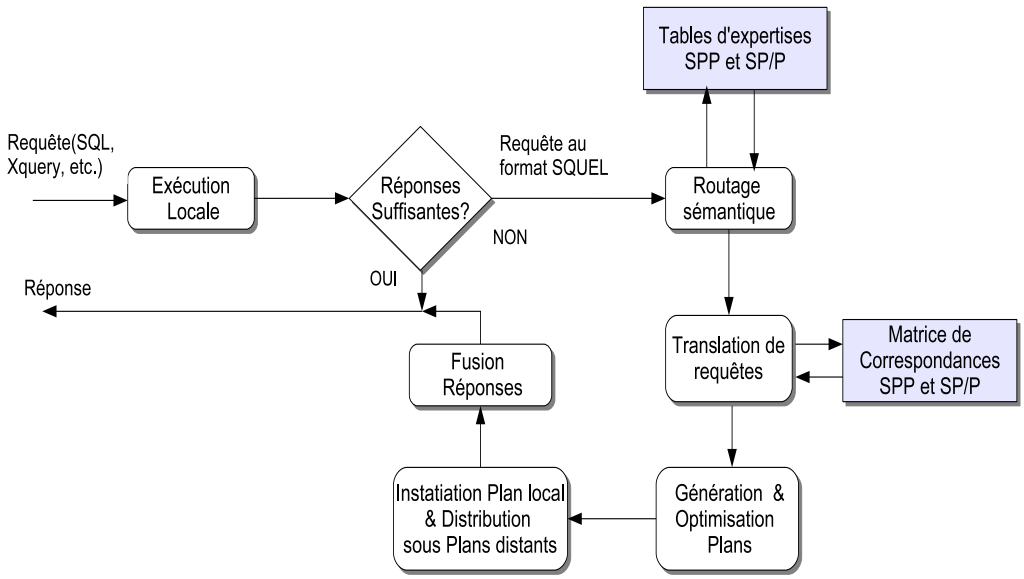


Figure 4.1 – Principales étapes du processus d'interrogation.

4.2.2 Sémantique de l'interrogation

Nous pouvons formaliser la sémantique des requêtes comme suit. Supposons un PDMS $\mathfrak{P} = (\mathcal{G}, \mathcal{M})$. Une requête Q est initiée par un pair P_i dans une communauté sémantique C_j et sur son propre schéma S_i. Les réponses attendues ne sont pas seulement des données $Q(\mathcal{D})$ incluses dans l'instance $I(S_i)$ du schéma S_i, mais aussi les données des pairs distants appartenant à la même communauté ou dans les communautés alliées. Pour interroger les pairs distants, la requête est envoyée au parrain super-pair de la communauté sémantique C_j qui va ensuite décomposer la requête en un ensemble de requêtes $\{Q^{(1)}, \ldots, Q^{(p)}\}(p < |\mathcal{SP} \cup \mathcal{P}|)$ où chaque requête $Q^{(k)}$ est définie sur le schéma S_k d'un pair prometteur pour la requête ou sur le *schéma suggéré* SS_k du parrain d'une communauté sémantique alliée. Le parrain fait suivre la requête aux (super-)pairs via une reformulation sémantique guidée par les correspondances sémantiques $\mathcal{M}$. Dans ce cas, la réponse à la requête est l'ensemble $\{Q^{(1)}(\mathcal{D}_{i_1}), \ldots, Q^{(p)}(\mathcal{D}_{i_p})\}$ avec $Q^{(k)}(\mathcal{D}_{i_k})$ un sous ensemble des données $I(S_k)$ du pair P_k ou un sous ensemble des données fournies par les pairs de la communauté C_k.

4.2.3 Format d'échange de requêtes SQUEL

Quand un pair formule une requête avec son langage d'interrogation (SQL, XQuery, etc.) celle-ci est d'abord exécutée localement puis réécrite dans le formalisme d'échange de requêtes SQUEL(*SenPeer Query Exchange Language*) et enfin envoyée à son super-pair pour pouvoir interroger les pairs distants sans connaître leur langage d'interrogation mais aussi pour éviter les multiples réécritures entre langages de requêtes différents. Une requête SQUEL est un arbre constitué de deux sous-arbres : sous arbre-condition et sous-arbre résultat. De plus, pour faciliter la découverte des pairs pertinents pour la requête nous associons à chaque nœud racine, attribut ou sous-requête l'ensemble des synonymes du nœud correspondant sémantiquement dans le $sGraph$ du pair initiateur de la requête. Il est important de noter que tout nœud attribut, racine ou sous-requête mentionné dans une requête SQUEL doit être défini dans le $sGraph$ associé.
Si par exemple le pair SAED de la figure 3.12 exprime la requête SQL suivante :

```
SELECT nom_sol,taux_sel,humidite
FROM   culture,sol
WHERE culture.nom_sol=sol.nom AND nomC='riz'
```

La requête enrichie exprimée en SQUEL sera alors celle représentée par la figure 6.
La définition de la classe représentant un nœud d'une requête SQUEL est donnée dans la Table 4.1.
Les requêtes SQUEL échangées à travers le réseau sont représentées sous forme de messages XML (voir annexes).

4.2.4 Navigation dans un graphe de requête SQUEL

Dans le but de naviguer dans un graphe de requête SQUEL, nous définissons un ensemble d'opérations qui nous permettent d'identifier les concepts représentant les nœuds importants pour la reformulation des requêtes. Nous introduisons d'abord les notations suivantes :

- $Node(c)$ est le nœud étiqueté par le concept c ;
- $Child(n)$ dénote un fils du nœud n ;
- $Parent(n)$ représente un nœud parent de n .

Soit $n_1 = Node(c_1)$ et $n_2 = Node(c_2)$, les opérations sont formulées comme suit :

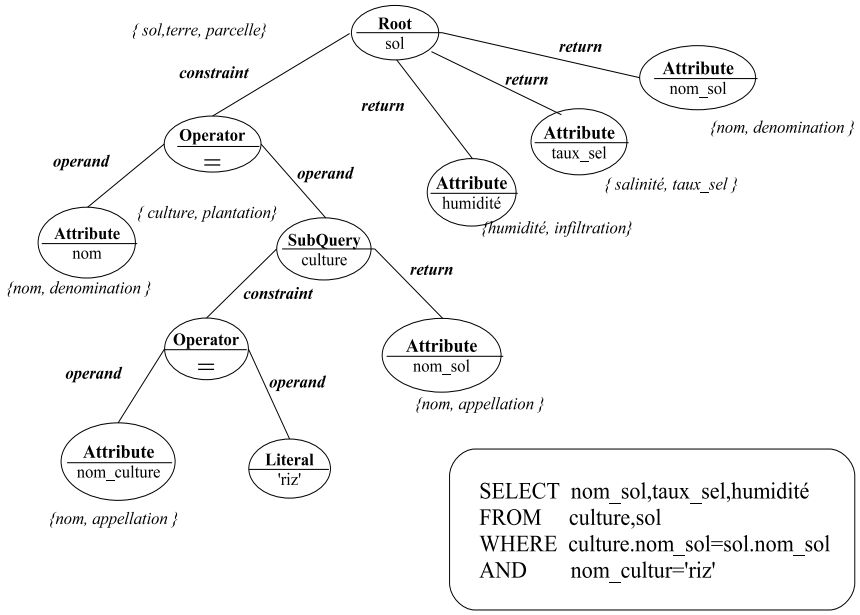


Figure 4.2 – Une requête SQL et la requête SQUEL correspondante.

$< QNode >$ ::= class **{**
$name :< word >$
$synonym : \{\}|\{< syn - set >\}$
$type :< typeNode >$
$constraint :< constraint >$
$return :< return >$
$operand :< operand >$**}**
$< syn - set >::=< word > \mid < word >, < synset >$
$< typeNode >::= \{\} \mid root \mid subQuery \mid operator \mid function \mid attribute \mid literal$
$< result >::= \{\} \mid \{< QNode >\}$
$< constraint >::= \{\} \mid \{< QNode >\}$
$< operand >::= \{\} \mid \{< QNode >\}$

Table 4.1 – Définition de la classe représentant un nœud du format d'échange de requêtes.

- $inComing$

$$n_2 \in inComing(n_1, r) \text{ ssi } n_2 = Parent(n_1) \wedge (n_2, n_1) = r. \tag{4.1}$$

  De façon informelle, cette relation retourne les nœuds parents d'un nœud n en suivant un arc de type r

- $outGoing$

$$n_2 \in outGoing(n_1, r) \text{ ssi } n_2 = Child(n_1) \wedge (n_1, n_2) = r. \tag{4.2}$$

  Cette relation retourne les nœuds fils d'un nœud n en suivant un arc de type r

A partir de ces opérations de navigation, nous définissons les prédicats suivants :

$$Type(n_1, n_2) = \begin{cases} vrai & \text{Si } outGoing(n_1, \texttt{"typeOf"}) = n_2 \wedge \\ & n_1 \in inComing(n_2, \texttt{"typeOf"}) \\ faux & \text{Sinon} \end{cases} \quad (4.3)$$

$$Return(n_1, n_2) = \begin{cases} vrai & \text{Si } n_2 \in outGoing(n_1, \texttt{"return"}) \wedge \\ & inComing(n_2, \texttt{"return"}) = n_1 \\ faux & \text{Sinon} \end{cases} \quad (4.4)$$

$$Constraint(n_1, n_2) = \begin{cases} vrai & \text{Si } outGoing(n_1, \texttt{"constraint"}) = n_2 \wedge \\ & inComing(n_2, \texttt{"constraint"}) = n_1 \\ faux & \text{Sinon} \end{cases} \quad (4.5)$$

$$Operand(n_1, n_2) = \begin{cases} vrai & \text{ssi } n_2 \in outGoing(n_1, \texttt{"operand"}) \wedge \\ & inComing(n_2, \texttt{"operand"}) = n_1 \\ faux & \text{sinon} \end{cases}$$

4.3 Mécanisme de routage sémantique

Dans ce paragraphe, nous proposons un nouveau algorithme de routage sémantique dans le contexte de SenPeer[36]. Dans un système P2P permettant l'interrogation de données structurées, un défi majeur est l'identification des pairs contenant des données pertinentes pour une requête. L'approche la plus fréquente pour identifier ces pairs est de déterminer si les entités et attributs de la requête apparaissent dans les sources de données interrogées. Les systèmes P2P non-structurés utilisent l'inondation et des approches aléatoires pour localiser des données, ce qui aboutit à beaucoup de traffic réseau, mais aussi des taux de rappel et de précision faibles. Pour améliorer leurs performances, le regroupement sémantique de pairs, mais aussi l'indexation permettent de sélectionner depuis un index les pairs pertinents à qui faire suivre une requête. Dans certains PDMS supportant des requêtes complexes, la sélection des pairs repose sur la description sémantique des pairs. Nous observons qu'elles sont essentiellement basées sur des observations statistiques et exploitent, dans certains cas, une ontologie commune ou une taxonomie. Dans certains cas, l'inondation semble inévitable.

Notre mécanisme de routage peut être appliqué en présence de plusieurs schémas/ontologies potentiellement différents, et pas seulement en présence d'un schéma partagé. D'autre part, la technique de routage combine la topologie sémantique induite par la réconciliation sémantique et les expertises des (super-)pairs pour propager efficacement les requêtes aux pairs pertinents dans une communauté mais aussi vers d'autres communautés. Le procédé de routage peut être appliqué en présence de plusieurs modèles de données, pourvu qu'il y ait des adaptateurs adéquats qui prennent en charge la transformation de la requête dans un langage adéquat pour chaque source de données. Le routage sémantique de requêtes est fait exclusivement par les super-pairs tandis que l'exécution des requêtes peut impliquer aussi bien les super-pairs que les pairs simples.

4.3.1 Sélection basée sur l'expertise

La sélection des pairs pertinents est basée sur leurs capacités à prendre en compte les éléments mentionnés dans la requête. Une façon naturelle d'évaluer cette capacité serait de comparer directement le graphe de la requête avec les $sGraph$ des voisins sémantiques. Étant donné que la comparaison entre graphe serait coûteuse en temps, nous ramenons les deux graphes à des structures beaucoup plus simples

à comparer. D'autre part, l'expertise constitue déjà un résumé du $sGraph$ du pair. Il reste donc à ramener le graphe de la requête dans le même formalisme. Pour cela, nous introduisons la notion de sujet de requête.

4.3.1.1 Sujet d'une requête.

Étant donné une requête, le super-pair doit être en mesure d'extraire son sujet en accord avec le $sGraph$ du (super-)pair ayant envoyé la requête. Le sujet d'une requête est une abstraction de la requête en termes des nœuds de l'arbre de la requête faisant référence à des éléments du $sGraph$ du (super-)pair ayant envoyé la requête.

$$Sub(Q) = \{(n_q, m_q) \in Q \mid Return(n_q, m_q)\} \cup \{(n_q, m_q) \in Q \mid \exists\, o_q \in Q \wedge Constraint(n_q, o_q) \wedge Operand(o_q, m_q)\}. \tag{4.6}$$

Par exemple, le sujet de la requête $Q1$ dans la figure 6 est :

```
Sub(Q1)={(sol,nom),(sol, taux_sel),(sol,humdite),(culture,nomC),
         (culture, nomsol)}.
```

Le sujet peut être vu comme complémentaire à l'expertise étant donné qu'il spécifie l'expertise nécessaire pour répondre à la requête. De plus, il peut être extrait automatiquement de la requête en considérant les schémas reçus par le super-pair.

4.3.1.2 Fonction de similarité sémantique

Une fois les phases d'abstraction de $sGraph$ en expertise et de requête SQUEL en sujet terminées, les deux structures résultantes sont mises en correspondance grâce à une fonction $Cap : Q \times P \rightarrow [0, 1]$ qui quantifie la capacité d'un pair P d'expertise $Exp(P)$ à répondre à une requête Q de sujet $Sub(Q)$.

$$Cap(P, Q) = \frac{1}{|Sub(Q)|}\left(\sum_{s \in Sub(Q)} \max_{e \in Exp(P)} S_s(s, e)\right). \tag{4.7}$$

La fonction Cap est basée sur la fonction S_s définie dans le chapitre précédent et qui détermine la similarité de deux ensembles de synonymes. Elle parcourt tous les couples du sujet et fait la moyenne de leurs similarités avec le couple le plus similaire dans l'expertise.

Un (super-)pair P peut prendre en charge une requête Q si la similarité entre son expertise et le sujet de la requête est supérieure à un seuil de similarité acceptable ϵ_{acc}:

$$map(P, Q) = \begin{cases} 1 & \text{Si } Cap(P, Q) \geq \epsilon_{acc} \\ 0 & \text{Sinon} \end{cases} \tag{4.8}$$

Notons que ces descriptions d'expertise et de sujet sont uniquement utilisées pour faciliter la découverte des pairs pertinents et donc elles n'altèrent pas le processus de traitement des requêtes. En effet, après la sélection des pairs pertinents, le super-pair revient à la requête SQUEL avec sa structure et ses contraintes pour la réécrire, en accord avec les matrices de correspondance, vers les pairs qu'il a réussi à détecter comme pertinents dans cette phase de sélection.

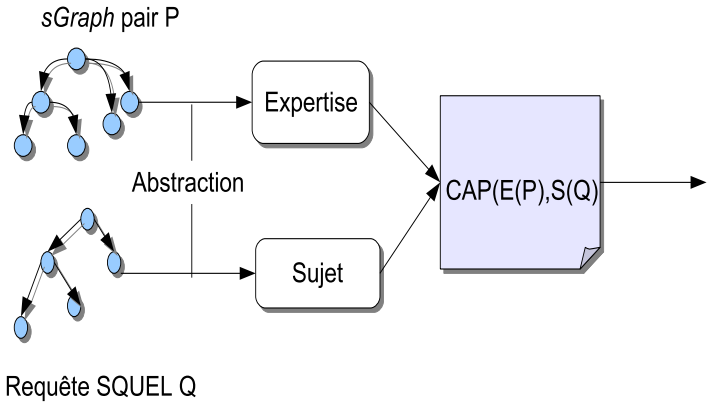


Figure 4.3 – Sélection basée sur l'expertise.

4.3.2 Algorithme de routage sémantique

L'algorithme de routage sémantique fournit une annotation sémantique de la requête avec les (super-)pairs pertinents. Cette requête annotée sera ensuite utilisée pour générer les plans de requêtes distribuées avant d'envoyer effectivement ces plans vers les pairs pertinents. En définitive, c'est une annotation sémantique suivie d'un routage physique vers les pairs sélectionnés. L'algorithme exploite la topologie sémantique induite par les correspondances sémantiques et les tables d'expertises des super-pairs pour propager la requête aussi bien dans une communauté sémantique que vers les communautés sémantiques associées. En effet, l'utilisation d'un catalogue sémantique pour déterminer les sources de données pertinentes pour une requête est une composante essentielle dans le traitement efficace des requêtes dans un environnement distribué. Supposons qu'un pair P_i de *parrain* super-pair SP_j, initie une requête Q sur son schéma avec son propre langage d'interrogation, nous distinguons les principales étapes suivantes : *décomposition de la requête en sous requêtes primitives, sélection des pairs pertinents pour chaque sous requête et annotation de ces sous requêtes* :

Algorithme 3 Routage sémantique de requête(Sélection des pairs)

Input : Requête SQUEL Q.
Output : Requête SQUEL annotée $AQ = \{< Q_x, P_{i_1}, \ldots, P_{i_k}, SP_{j_1}, \ldots, SP_{j_l} >\}$
$SQ := queryParts(Q)$
$AQ := \{\}$
Pour tout $(super-)pair R_j \in \mathcal{P} \cup \mathcal{SP}$ **Faire**
  Si $Cap(Sub(Q), Exp(R_j) > \epsilon_{acc}$ **Alors**
    $AQ := AQ \cup \{< Q, R_j >\}$
  Sinon
    Pour tout $Q_i \in SQ$ **Faire**
      Si $Cap(Sub(Q_i), Exp(R_j) > \epsilon_{acc}$ **Alors**
        $AQ := AQ \cup \{< Q_i, R_j >\}$
      Fin Si
    Fin Pour
  Fin Si
Fin Pour

L'algorithme d'annotation commence d'abord par décomposer, si nécessaire les requête composites en requêtes primitives. Nous rappelons qu'une requête primitive ne contient pas de nœud sous-requête.

Ensuite, le sujet de la requête SQUEL est extrait et mis en correspondance par le biais de la fonction de similarité Cap, avec les expertises des (super-)pairs présentes dans les tables d'expertises $E_{SP/SP}$ et $E_{SP/P}$. Ainsi donc, tous les pairs dans la table d'expertise ayant une capacité supérieure au seuil acceptable sont utilisés pour annoter la requête. Si un pair ne répond pas entièrement à la requête, l'algorithme vérifie s'il peut prendre en charge des sous requêtes moins complexes et si tel est le cas, ces dernières sont annotées avec le pair en question.

La requête annotée sera plus tard passée au générateur de plans pour la construction de plans de requêtes prenant en compte la distribution de l'information que vient de découvrir le routage sémantique de requête.

4.3.3 Illustration

Pour illustrer le routage sémantique, considérons que pair P_{SAED} dans la Figure 3.12 envoie à son *parrain* super-pair SP_A la requête Q décrite dans la figure 6. SP_A utilise la topologie sémantique illustrée par la figure 3.14 pour router la requête. A cet effet, SP_A évalue la capacité de chacun de ses voisins sémantiques P_{ISRA} et SP_P . Les valeurs suivantes de capacités sont renvoyées $cap(Sub(Q), Exp(P_{ISRA})) = 0.8$ et $cap(Sub(Q), Exp(SP_P)) = 0.65$. Sur la base de tels résultats SP_A envoie la requête Q aux voisins sémantiques sélectionnés avant d'attendre leurs réponses. La requête annotée est représentée graphiquement par la figure 4.4

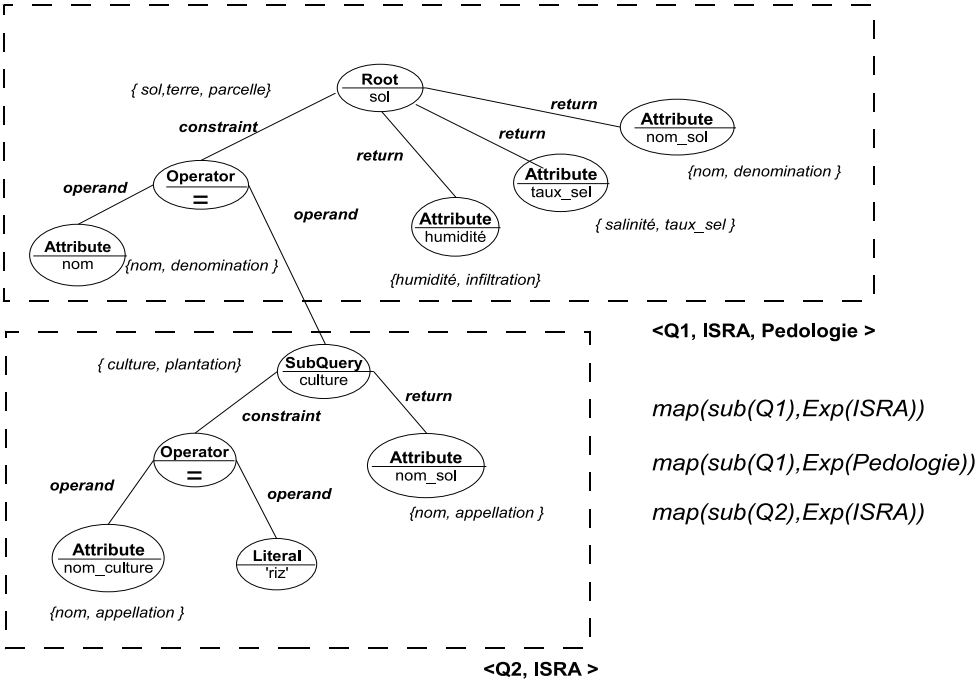


Figure 4.4 – Annotation sémantique de requête.

4.4 Reformulation de requêtes

Dans le chapitre précédent, nous avons introduit le processus de découverte des correspondances sémantiques entre deux pairs en accord avec leurs schémas. Dans ce paragraphe, nous traitons le problème de la reformulation de requêtes SQUEL entre pairs distants. Nous montrons comment réécrire une requête SQUEL, exprimée sur un schéma source, en une autre requête sur un schéma cible, en accord avec les correspondances sémantiques, de telle sorte que la requête réécrite sur les données du pair cible retourne des réponses sémantiquement correctes.

4.4.1 Le problème de la reformulation de requête

La reformulation de requête constitue un problème complexe à cause de la diversité des connaissances sémantiques qui doivent être prises en compte pour assurer la validité de la reformulation. Le but principal de la reformulation de requête est de réécrire une requête de pair en un ensemble de requêtes équivalentes pouvant être évaluées sur des pairs distants, chacune correspondant à une manière alternative de calculer le résultat. A cet effet, l'utilisation des correspondances sémantiques permet de générer des réécritures consistantes de la source vers la cible en considérant les contraintes et la structure du schéma cible. Le problème de réécriture pris en charge par les super-pairs peut être défini comme suit :

Définition 4.1. Étant donné un pair P formulant une requête Q, avec un vocabulaire $\mathcal{V}(Q)$, sur un schéma source S, comment trouver une requête reformulée Q^r de Q, avec un vocabulaire $\mathcal{V}(Q^r)$, sur un schéma cible T de telle sorte que Q^r retourne des résultats sémantiquement équivalents aux résultats de Q. Le vocabulaire de la requête est l'ensemble des nœuds originaires du schéma et mentionnés dans les parties *résultat* et *contrainte* de la requête mais aussi les littéraux (données).

4.4.2 Algorithme de reformulation de requête

L'algorithme de reformulation de requête est défini en accord avec la sémantique formelle des requêtes. La reformulation renomme les nœuds de Q originaires du schéma source S en des nœuds provenant du schéma cible T, par le biais de la matrice de correspondance $\mathcal{M}_{S\leftrightarrow T}$. Au début du processus de reformulation, toutes les requêtes sont primitives.

Définition 4.2 (Projection de schéma)**.** Soit S un schéma source et T un schéma cible. La projection de S sur T, notée $\Pi_T(S)$, est constituée de l'ensemble des nœuds de S ayant une correspondance non nulle dans T.

$$\Pi_T(S) = \{n^S \in S \mid \exists\, n^T \in T \wedge \mathcal{M}_{S\leftrightarrow T}(n^S, n^T) \neq null\}. \tag{4.9}$$

$\Pi_T(S)$ est aussi appelé schéma de S *par rapport à* T.

La reformulation prend en compte deux niveaux de correspondances. Ceci implique que le traitement de la requête peut être réalisé en termes de deux phases de reformulation : reformulation dans une communauté et reformulation vers les communautés alliées. Ces deux phases de reformulation sont similaires. Elles sont constituées des étapes suivantes :

- *Analyse syntaxique*
- *Détermination du nœud racine*
- *Détermination du résultat*
- *Reformulation des contraintes atomiques*
- *Phase de fusion*

Soit E une expression et $E[a \rightarrow b]$ l'expression E dans laquelle on a substitué toutes les occurrences de a dans E par b. Dans la suite, nous présentons notre algorithme de reformulation organisé en cinq étapes.

Étape 1. *Analyse syntaxique*. Chaque requête SQUEL Q est parcourue pour identifier les composantes particulières de la requête, contraintes, nœud racine et liste de résultat. De plus, toutes les assertions de correspondances pertinentes pour les nœuds de la requête sont identifiées. Après la traversée de la requête, cette dernière est associée au triplet (Q_{R0}, Q_{RE}, Q_{CT}) où :

- $Q_{R0} = \{n^S \in S \,|\, Type(n^S, ROOT)\}$ contient le nœud racine de la requête;
- $Q_{RE} = \{n^S \in S \,|\, \exists\, m^S \in S \wedge Return(n^S, m^S) \wedge Type(m^S, ROOT)\}$ contient les nœuds dans la partie *return* du graphe de requête ;
- Q_{CT} est l'arbre de contraintes de la requête initiale avec des nœuds internes représentant des opérateurs booléens-conjonction, disjonction et négation (AND, OR, NOT) ou des opérateurs relationnels ($<, >, \leq, \geq, =, \neq$) et dont les feuilles sont des éléments issus du schéma. Chaque sous arbre avec comme nœud interne un opérateur relationnel représente une contrainte atomique dans l'ensemble $\{n_i^S\ \theta\ n_j^S \,|\, (n_i^S\, n_j^S, \theta) \in S^3 \wedge operand(\theta, n_i^S) \wedge operand(\theta, n_j^S)\} \cup \{n_i^S\, \theta\, c\}$ avec $\theta \in \{<, >, \leq, \geq, =, \neq\}$ et c une valeur constante.

Ce triplet (Q_{R0}, Q_{RE}, Q_{CT}) est l'entrée de l'algorithme de reformulation et sera réécrit en un autre triplet $(Q^r_{R0}, Q^r_{RE}, Q^r_{CT})$ représentant la requête cible.

Étape 2. *Détermination du nœud racine*. Le nœud racine résultat dans Q^r_{R0} est déterminé en considérant les éléments dans T avec une correspondance non nulle avec le nœud racine en entrée dans Q_{R0}. Si un tel nœud n'existe pas, alors la reformulation échoue. Dans le cas contraire, le nœud racine résultat Q^r_{R0} est déterminé à partir du nœud racine en entrée Q_{R0} comme suit :

$$Q^r_{R0} = Q_{R0}[n^S \rightarrow n^T], n^S \in \Pi_T(S) \wedge TypeOf(n^S, ROOT) \wedge \mathcal{M}_{S \leftrightarrow T}(n^S, n^T) \neq null\}. \tag{4.10}$$

Étape 3. *Détermination du résultat* La liste de retour Q^r_{RE} de Q^r est calculée en considérant les nœuds dans la liste de retour Q_{RE} en entrée ayant une correspondance non nulle dans S. La liste de retour Q_{RE} en entrée est donc transformée en une liste de retour en sortie Q^r_{RE} sur la base de la matrice de correspondance comme suit :

$$Q^r_{RE} = \bigcup_{n^S \in \Pi_T(S) \wedge \mathcal{M}_{S \leftrightarrow T}(n^S, n^T) \neq null\}} Q_{RE}[n^S \rightarrow n^T]. \tag{4.11}$$

Étape 4. *Reformulation des contraintes atomiques* Chaque sous arbre représentant une contrainte atomique est reformulée, sur la base de la matrice de correspondances $\mathcal{M}_{S \leftrightarrow T}$, en une contrainte pouvant être évaluée sur le schéma cible T comme suit :

- une contrainte atomique $(n^S\, \theta\, c)$ est reformulée en :

$$(n^S\, \theta\, c) \rightarrow \begin{cases} (n^T\, \theta\, c) & \text{Si} \quad \mathcal{M}_{S \leftrightarrow T}(n^S, n^T) \neq null \\ vrai & \text{Sinon} \end{cases} \tag{4.12}$$

- une contrainte atomique $(n_i^S\, \theta\, n_j^S)$ est reformulée en

$$(n_i^S\, \theta\, n_j^S) \rightarrow \begin{cases} (n_k^T\, \theta\, n_p^T) & \text{Si } \mathcal{M}_{S \leftrightarrow T}(n_i^S, n_k^T) \neq null \quad \text{et} \quad \mathcal{M}_{S \leftrightarrow T}(n_j^S, n_p^T) \neq null \\ vrai & \text{Sinon} \end{cases} \tag{4.13}$$

Étape 5. *Phase de fusion*. Cette phase fait référence à la finalisation de la requête SQUEL cible sur la base du triplet $(Q^r_{R0}, Q^r_{RE}, Q^r_C)$ généré précédemment. A partir de cet ensemble le graphe de requête SQUEL Q^r correspondant est construit.

Nous précisons que le processus de réécriture de requêtes ne concerne que les requêtes primitives. Après la phase de réécriture des requêtes primitives, tous les fragments de requêtes qui ont été séparés avant le processus de reformulation doivent être recomposés.

4.4.3 Illustration

Pour illustrer la reformulation de requête, supposons que le pair P_{SAED} dans le PDMS indiqué dans la figure 3.12 exprime la requête Q décrite dans la figure 6. La requête peut être interprétée comme suit :

"Trouver les caractéristiques des sols sur lesquels le riz est cultivé ".

La requête est ensuite envoyée au *parrain* super-pair SP_A qui la réécrit dans le vocabulaire de la communauté et la décompose en deux requêtes primitives Q_1 et Q_2, à cause de la présence d'un nœud sous-requête. Après la phase de routage sémantique SP_A annote chaque sous requête réécrite comme suit :

$[Q1, SP_P, P_{ISRA}]$ et $[Q2, P_{ISRA}]$.

Les triplets résultant de l'analyse syntaxique sont :

$Q_1 : (< sol >, < nomSol, salinite, humidite >)$.

$Q_2 : (< culture >, < nom, nom\_sol >, < nom =' riz' >)$.

La phase suivante est la reformulation en utilisant les correspondances sémantiques trouvées durant la formation de la topologie sémantique. Cette phase inclue les déterminations du nœud racine et de la liste de retour mais aussi la matérialisation des contraintes atomiques. En tenant compte des correspondances sémantiques, les reformulations résultantes sont alors :

$Q^r_1 \rightarrow SP_P : (< terre >, < nomTerre, taux\_sel, infiltration >)$.

$Q^r_1 \rightarrow P_{ISRA} : (< terre >, < nomTerre, taux\_sel, hygrometrie >)$.

$Q^r_2 \rightarrow P_{ISRA} : (< culture >< nomTerre >, < nomC = "riz" >)$.

Ensuite, le parrain initie la phase de fusion pour construire les sous requêtes résultantes.

4.4.4 Attributs manquants

En réécrivant une requête, une perte d'information peut arriver. Une requête reformulée de la requête source peut ne pas être envoyée à un (super-)pair si ce dernier ne dispose pas d'assez d'attributs significatifs pour répondre à la requête sur sa source de données ou dans la communauté sémantique pour les raisons suivantes :

- il n'y a pas de correspondance sémantique dans la matrice de correspondance pour réécrire le nœud racine de la requête, donc l'absence de cet attribut significatif rend impossible la construction d'une requête valide ;
- il n' y a pas d'assertion de correspondance pour aider à la reformulation des contraintes atomiques. Par conséquent, l'absence des ces attributs significatifs dans la partie contrainte de la requête peut changer le sens de la requête.

Les attributs manquants dans les contraintes atomiques peuvent changer le sens de la requête. La définition d'*inclusion* de requête permet de savoir si un attribut ou une condition manquante peuvent changer le sens de la requête. Nous voulons des requêtes réécrites qui soient contenues dans la requête source de telle sorte que les réponses renvoyées soient un sous-ensemble de celles attendues. Étant donné que nous

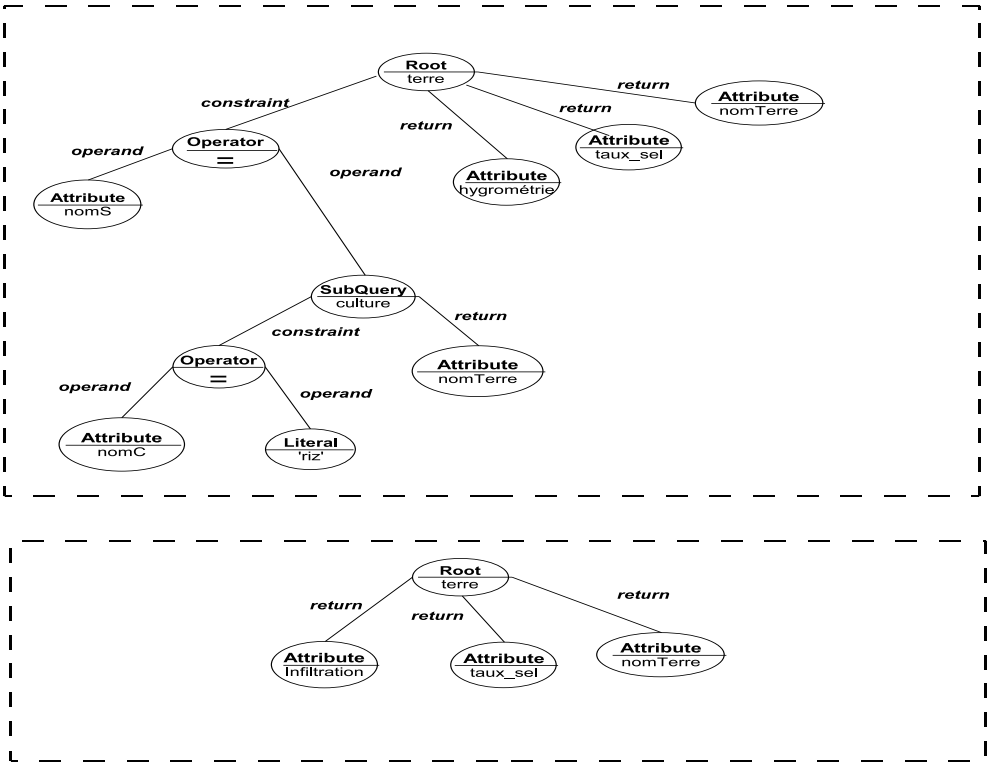


Figure 4.5 – Requêtes réécrites.

voulons plus de réponses correctes ou approximatives, une contrainte atomique est évaluée comme étant *vraie* dans le cas où il y a un attribut manquant dans la contrainte. La performance du PDMS peut être améliorée en envoyant les requêtes aux (super-)pairs pour lesquels on peut prédire qu'ils seront en mesure de répondre à la requête, en accord avec les attributs manquants. A cet effet, une requête reformulée est envoyée à un (super-)pair si

(i) ce dernier contient tous les attributs dans la requête source, ou

(ii) il y a des attributs manquants dans la partie résultat de la requête ou dans les contraintes atomiques mais sans être contradictoires avec les conditions de la requête source.

4.4.5 Inclusion de requêtes

L'inclusion de requêtes vérifie si les résultats d'une requête sont contenues dans les résultats d'une autre requête et vice-versa[77]. Dans notre contexte, nous nous intéressons à deux types d'inclusions. La première, appelée *inclusion locale* vérifie si une requête reformulée de pair est contenue dans une union de requêtes locales $Q_1, \ldots, Q_n$ dans une communauté sémantique. Le second type d'inclusion, appelé *inclusion distante* est à propos de l'inclusion entre une requête Q^r sortant d'une communauté et une requête Q^i dans une autre communauté sémantique. La contenance est basée sur la notion de réponse certaine (*certain answer*)[42]. Le concept de réponse certaine a été introduit dans le traitement des requêtes basées sur les vues. Nous l'adaptons à notre contexte en définissant le concept de réponse certaine dans une communauté appelé *réponse certaine locale* et de réponse certaine d'une communauté alliée appelée *réponse certaine distante*. Avant les définitions de réponse certaine locale et de réponse

certaine distante, nous introduisons les définitions primaires suivantes.

Définition 4.3. (*Source de données induite d'une communauté*) Étant donné une communauté sémantique C avec un schéma suggéré SS, un ensemble de pairs affiliés de schémas $S_1, \ldots, S_n$ respectivement associés aux sources de données $I(S_1), \ldots, I(S_n)$ et une matrice de correspondance intra-communauté $\mathcal{M}_{SP/P}$, la source de données induite $\mathbb{D}$ de la communauté C est telle que l'assertion $\varphi(SS) = \varphi(S_1) \cup \ldots \cup \varphi(S_n)$ est vérifiée sur $I(S_1), \ldots, I(S_n)$.

Définition 4.4. (*Source de données projetée*) Étant donné une communauté sémantique C_k avec un schéma suggéré SS_k, un ensemble de super-pairs alliés avec des schémas suggérés $SS_1, \ldots, SS_m$ et une matrice de correspondance inter-communauté $\mathcal{M}_{SP/SP}$, la source de données projetée $\mathbb{B}_k$ de la communauté par rapport à toutes les sources de données des communautés alliées est celle pour laquelle l'assertion $\varphi(SS_k) = \bigcup_{(1 \leq i \leq m, i \neq k)} \varphi(SS_i)$ est vérifiée sur les sources de données induites $\mathbb{D}_1, \ldots, \mathbb{D}_m$ des communautés alliées $C_1, \ldots, C_m$.

Nous pouvons, à présent, définir le concept de réponse certaine (locale et distante) en nous basant sur les concepts de source de données induite et de source de données projetée.

Définition 4.5. (*réponse certaine locale*) Soit C une communauté sémantique avec un schéma suggéré SS et un ensemble de pairs affiliés de schémas $S_1, \ldots, S_n$ respectivement associés aux sources de données $I(S_1), \ldots, I(S_n)$. Soit $\mathcal{M}_{SP/P}$ la matrice de correspondance intra-communauté, Q une requête réécrite sur le schéma suggéré SS. La *réponse certaine locale* à Q en accord avec $I(S_1), \ldots, I(S_n)$ et sur la base de $\mathcal{M}_{SP/P}$ est le résultat de l'évaluation de Q sur la base de données induite $\mathbb{D}$ de la communauté C, dénotée $Cert_{\mathcal{M}}(Q) = Q(\mathbb{D})$.

Définition 4.6. (*réponse certaine distante*) Étant donné une communauté sémantique C_k avec un schéma suggéré SS_k, un ensemble de super-pairs alliés avec des schémas suggérés $SS_1, \ldots, SS_m$. Soit $\mathcal{M}_{SP/SP}$ la matrice de correspondance inter-communautés, Q_k une requête réécrite sur le schéma suggéré SS_k; La réponse certaine distante à Q_k en accord avec les bases de données induites $\mathbb{D}_1, \ldots, \mathbb{D}_m$ des communautés sémantiques $C_1, \ldots, C_m$ sur la base de $\mathcal{M}_{SP/SP}$ est le résultat de l'évaluation de Q_k sur la base de données projetée $\mathbb{B}_k$ de SS_k, dénotée $Cert_{\mathcal{M}}(Q_k) = Q(\mathbb{B}_k)$.

Nous pouvons maintenant définir l'inclusion de requêtes dans notre contexte.

Définition 4.7. (*inclusion locale de requête*) Soit C une communauté sémantique de schéma suggéré SS, avec un ensemble de pairs de schémas $S_1, \ldots, S_n$ et $\mathcal{M}_{SP/P}$ la matrice de correspondance intra-communauté. Soit Q une requête au sein de la communauté et $Q^u = Q_1 \cup \ldots \cup Q_n$ une union de requêtes locales à la communauté. Q est dite être *localement contenue* dans Q^u (notée $Q \subseteq_{\mathcal{M}} Q^u$) dans la communauté sémantique C si pour toutes les sources de données des pairs $I(S_1), \ldots, I(S_n)$ nous avons $cert_{\mathcal{M}}(Q) \subseteq Q_1(I(S_1) \cup \ldots \cup Q_n(I(S_n))$. Si $Q \subseteq_{\mathcal{M}} Q^u$ et $Q \supseteq_{\mathcal{M}} Q^u$ nous avons une équivalence locale notée $Q \equiv_{\mathcal{M}} Q^u$.

Définition 4.8. (*inclusion distante*). Soit C une communauté sémantique de schéma suggéré SS, avec $SS_1, \ldots, SS_m$ les schémas suggérés des communautés alliées et $M_{SP/SP}$ la matrice de correspondance inter-communauté. Soit Q une requête de pair réécrite sur SS et Q_k^r une requête distante sur un schéma suggéré d'une communauté alliée. Q est dite *incluse de façon distante* dans Q_k^r (notée $Q \subseteq_{\mathcal{M}} Q_k^r$) si pour les deux sources induites $\mathbb{D}$ et $\mathbb{D}_k$ des communautés, nous avons $cert_{\mathcal{M}}(Q) \cup Q(\mathbb{D}) \subseteq cert_{\mathcal{M}}(Q_k^r) \cup Q_k^r(\mathbb{D}_k)$. En cas d'inclusion distante dans les deux directions, c'est-à-dire $Q \subseteq_{\mathcal{M}} Q_k^r$ et $Q \supseteq_{\mathcal{M}} Q_k^r$ nous disons que Q et Q_k^r sont *équivalentes de façon distante* et nous notons $Q \equiv_{\mathcal{M}} Q_k^r$.

4.4.6 Validité de la reformulation de requêtes

La validité assure que la requête réécrite retourne des résultats corrects depuis les sources de données des pairs sélectionnés comme pertinents pour la requête initiale. Étant donné que les pairs peuvent joindre ou quitter le système à tout moment, dans la plupart des cas, nous n'obtenons pas de réponse complète à la requête, c'est-à-dire, toutes les données pertinentes dans le réseau, mais au moins nous avons des réponses (valides) qui peuvent être considérées comme complètes en accord avec l'ensemble des pairs actifs.

Soit S et T deux $sGraph$ et $\mathcal{M}_{S\leftrightarrow T}$ la matrice de correspondance pour S et T. $\mathcal{M}_{S\leftrightarrow T}$ est associée avec la fonction de correspondance $\mu : S \rightarrow T$. Soit $I(S)$ and $I(T)$ les instances respectives des $sGraph$ S et T.

Définition 4.9. Soit Q^S une requête exprimée sur S et Q^T la reformulation de Q^S sur T. La reformulation Q^T est *correcte* par rapport à $\mathcal{M}_{S\leftrightarrow T}$, dénotée $Q^S \rightsquigarrow_{\mathcal{M}} Q^T$ si $\forall d \in I(S)\ Q^S(d) = Q^T(\mu(d))$, c'est-à-dire, Q^S et Q^T appliquée à l'instance transformée retournent le même résultat pour toute instance $d \in I(S)$. La *validité* de l'algorithme de réécriture est atteinte s'il peut trouver seulement les requêtes $Q^T(T)$ qui sont correctes. Les résultats sont *complets* s'il peut trouver toutes les requêtes $Q^T(T)$ correctes.

La reformulation Q^T appliquée à d_T est comme une composition de fonctions de la fonction μ de S à T, et de la fonction Q^S appliquée à d_S. Tout nœud α^S dans la racine, la partie résultat ou la partie contrainte de Q^S avec un élément correspondant β^T dans la matrice de correspondance est substitué par la variable nœud correspondante. Par le biais de ces correspondances et de la substitution nous pouvons obtenir le nœud racine, identifier les nœuds requêtes correspondants originaires de T et matérialiser les contraintes correspondantes. Pour ces raisons, nous pouvons affirmer que Q^T correspond bien à appliquer correctement les correspondantes sémantiques à Q^S. Donc, $\forall d \in I(S)$, Q^T appliquée à l'instance transformée $\mu(d)$ et Q^S appliquée à d renvoient le même résultat, c'est-à-dire $\forall d \in I(S)$: $Q^S(d) = Q^T(\mu(d))$.

4.5 Génération des plans de requêtes

4.5.1 Vue d'ensemble

La génération des plans de requêtes est responsable de la construction de plans de requêtes distribuées qui supervisent l'exécution des requêtes de façon distribuée en considérant tous les pairs contribuant au résultat final des requêtes. Chaque plan est partitionné à travers les (super-)pairs qui exécutent leur part de la requête. La génération des plans dans notre PDMS pose plusieurs défis. Nous pouvons distinguer trois approches principales quant à la génération des plans de requête à partir des informations de localisation des pairs renvoyées par l'algorithme de routage :

(i) Un seul nœud définit les plans des requêtes avec une vue globale du PDMS ;

(ii) Chaque pair, sur la base de sa connaissance locale, génère des plans de requêtes qui peuvent être incomplets ;

(iii) Un ensemble de pairs spéciaux, ayant des connaissances à propos de petits groupes de pairs, générent des plans adéquats impliquant ces derniers.

Étant donné que nous supposons un environnement distribué et donc, à cause du manque de connaissance globale (e.g. à propos de la distribution des données) la génération des plans est beaucoup plus

difficile que dans un système centralisé et l'approche centralisée n'est pas applicable. La deuxième méthode est adaptée à un environnement P2P complètement distribué. Toutefois, il semble invraisemblable que des résultats acceptables puissent être délivrés. Par conséquent, nous adoptons la troisième approche.

Comme nous l'avons dit auparavant, les plans ne peuvent pas être générés de façon statique par un nœud du PDMS contrairement aux techniques d'optimisation dans les bases de données distribuées traditionnelles. En effet, la table d'expertise qui guide le processus de distribution est dynamique et distribué entre super-pairs, et donc une optimisation statique n'est pas possible. Par conséquent, dans SenPeer les problèmes surgissant dans le traitement distribué des requêtes sont délaissés aux super-pairs.

Le générateur de plans met en place des plans de requêtes distribuées en accord avec les informations de localisation renvoyées par l'algorithme de routage sémantique. Ce module prend en entrée une requête de pair et génère en sortie un plan de requête logique qui indique comment la requête doit être répondue. Le plan de requête logique est constitué d'une ou de plusieurs sous-requêtes connectées via des opérateurs.

Étant donnée que la génération des plans est une tâche coûteuse, il serait bénéfique de considérer une stratégie de génération de plans qui assigne des fragments de plans à d'autres (super-)pairs impliqués dans la requête. Ces (super-)pairs devront coopérer durant le processus de compilation, étant donné qu'aucun d'entre eux n'a une connaissance complète du coût d'exécution et aussi parce qu'à cause de l'autonomie, aucun d'entre eux n'a un accès direct à tous les fragments du plan d'exécution de requête. Pour ce faire, chaque parrain super-pair recevant une requête provenant de sa communauté génère un plan d'exécution de requête qui supervise l'exécution de la requête, l'optimise partiellement et instancie les plans locaux pour les sous-requêtes qui peuvent être traitées au sein de la communauté et enfin envoie les sous-requêtes restantes vers ses alliées super-pairs mentionnés dans le plan de requête courant. Il est important de noter qu'à ce stade aucune génération de plan ou optimisation complète ne peut être opérée, étant donné que le parrain super-pair a une vue du PDMS limitée à sa communauté sémantique et à ses alliés.

Après la phase d'optimisation locale, le parrain instancie le plan de requête local en créant les canaux de communication appropriés entre lui et les pairs impliqués dans la requête. Le plan de requête local est exécuté dans la communauté dans le but d'obtenir les premières parties du résultat de la requête le plus vite possible. Étant donné que les pairs reçoivent des plans de requêtes logiques, ils se chargent seulement de les transformer en plans de requêtes physiques en décidant de l'ordonnancement des opérateurs. De plus, les sous-requêtes restantes sont distribuées aux super-pairs pertinents. Ces derniers vont continuer le processus d'optimisation sur des fragments de plans plus petits en appliquant successivement l'algorithme de routage dans leur communauté, la génération et l'optimisation des plans de requêtes correspondants, avant de les faire suivre aux pairs concernés dans leurs communautés. Par conséquent, l'évaluation du plan de requête complet est faite en plusieurs étapes en imbriquant les phases de routage et de génération de plans pour obtenir une réponse complète et correcte à la réponse. Cette imbrication favorise le parallélisme dans l'exécution de plusieurs parties du plan. Dès lors, le temps nécessaire pour recevoir les premiers résultats est plus petit et le traitement intra-pair est favorisé. Ceci est un aspect important dans les systèmes P2P, vu que beaucoup d'utilisateurs veulent recevoir la réponse à leurs requêtes le plus tôt possible. Il est certes possible d'envisager un scénario permettant une exécution séquentielle des phases de routage et de génération des plans avec à l'arrivée un seul plan de requête aboutissant à des résultats complets. Cependant l'utilisateur devra attendre que la phase de génération de plans, qui est assez coûteuse, se termine. Il devra aussi attendre que l'exécution de ce même plan, qui va consommer beaucoup de temps d'exécution, finisse.

Nous utilisons un modèle de requêtes à trois niveaux :

- au niveau supérieur nous avons les sous-plans ;

- au deuxième niveau nous avons les opérateurs virtuels connectant les sous-plans. Ces opérateurs sont virtuels en ce sens qu'ils ne sont pas encore annotés avec un (super-)pair spécifique ;
- le troisième niveau concerne les opérateurs concrets. Ce sont des opérateurs virtuels annotés avec les (super-)pairs spécifiques devant les exécuter.

4.5.2 Réécriture algébrique des requêtes

Les plans de requêtes sont souvent représentés par des arbres. Les nœuds de tels arbres correspondent aux opérateurs de la requête, tandis que les feuilles représentent les expressions de chemins, extraites du graphe de la requête et qui doivent être combinées en vue de produire le résultat final. Nous précisons que ces fragments de plans correspondent à des requêtes primitives, étant donné que l'algorithme de routage sémantique identifie les (super-)pairs qui peuvent répondre à la totalité d'une requête primitive. De plus, le plan de la requête spécifie, pour chaque opérateur de la requête, le (super)-pair qui devrait le traiter dans le but de combiner les résultats intermédiaires en vue d'obtenir une réponse finale. De la requête annotée retournée par l'algorithme de routage sémantique, une réécriture algébrique de la requête initiale est produite.

4.5.2.1 Opérateurs de l'algèbre

Un algèbre est une structure formelle constitué d'ensembles et d'opérations sur les éléments de ces ensembles. Par exemple, l'algèbre relationnel est un système formel pour manipuler les relations. Les opérandes de l'algèbre relationnel sont les relations. Les opérations incluent l'ensemble d'opérateurs traditionnels tels que la sélection, la projection et la jointure. Étant donné que les opérateurs algébriques ont une précédence et un ordre d'application, un algèbre de requêtes est procédural dans le sens qu'il prescrit comme construire le résultat d'une requête. D'autre part, des opérateurs de l'algèbre peuvent être implémentés par différents algorithmes, chacun avec un coût différent.

Pour notre algèbre nous utilisons les opérateurs courants pour représenter les requêtes SQUEL sous forme de plans de requête : union($\cup$), intersection($\cap$), différence($\setminus$), sélection(σ), jointure($\bowtie$), projection(π). Le tableau 4.2 décrit les opérateurs nous permettant d'exprimer une requête SQUEL sous forme algébrique.

Opération	Définition	Cardinalité
$\sigma_P(r)$	$\{t \mid t \in r \wedge P(t)\}$	$\leq \lvert r \rvert$
$\Pi_{B_1,\ldots,B_k}(r)$	$\{(w_1,\ldots,w_k \mid \exists t = (v_1,\ldots,v_n) \forall_{i=1}^{k} \exists j\, B_i = A_j \wedge w_i = v_j\}$	$= \lvert r \rvert$
$r_1 \cup r_2$	$\{t \mid t \in r_1 \vee t \in r_2\}$	$(\geq \lvert r_1 \rvert) \wedge (\leq \lvert r_1 \rvert + \lvert r_2 \rvert)$
$r_1 \cap r_2$	$\{t \mid t \in r_1 \wedge t \in r_2\}$	$\leq max(\lvert r_1 \rvert, \lvert r_2 \rvert)$
$r_1 \times r_2$	$\{(a_1,\ldots,a_m,b_1,\ldots,b_n \mid (a_1,\ldots,a_m) \in r_1 \wedge (b_1,\ldots,b_n) \in r_2\}$	$= \lvert r_1 \rvert \cdot \lvert r_2 \rvert$
$r_1 \bowtie_P r_2$	$\{t_1 \cdot t_2 \mid t_1 \in r_1 \wedge t_2 \in r_1 \wedge P(t_1,t_2)\}$	$=r_1 \mid * \mid r_2 \mid / selectiviteJoin$
$r_1 \setminus r_2$	$\{t \mid t \in r_1 \wedge t \notin r_2\}$	$(\geq \lvert r_1 \rvert - \lvert r_2 \rvert) \wedge (\leq \lvert r_1 \rvert)$

Table 4.2 – Liste des opérateurs algébriques

4.5.2.2 *Algorithme mise en forme algébrique*

L'algorithme de réécriture algébrique prend en entrée une requête SQUEL annotée et retourne en sortie le plan de requête correspondant. Dans un premier temps, la requête annotée est traversée et pour chaque sous-requête Q_x, l'ensemble RP_x des (super-)pairs pertinents est extrait des annotations. Le résultat, pour chaque sous-requête Q_x, est l'union des résultats $Q_x@P_y$ provenant de chaque pair P_y détenant des données pertinentes pour la requête ou $Q_i@SP_y$ provenant de chaque super-pair, parrain d'une communauté alliée ayant des pairs avec des données pertinentes, s'il y a plus d'un (super-)pair pertinent pour la sous-requête considérée. ($QEP_x = \left(\bigcup_{i=i_1}^{i=i_k} Q_x@P_i\right) \cup \left(\bigcup_{j=j_1}^{j=j_l} Q_x@SP_j\right)$. Le plan de

Algorithme 4 Mise en forme algébrique

Entrée : une requête annotée $AQ = \{< Q_x, P_{i_1}, \ldots, P_{i_k}, SP_{j_1}, \ldots, SP_{j_l} >\}$
Sortie : Un plan de requête QEP correspondant à AQ
$QEP := \{\}$
Pour tout $Q_x | \{< Q_x, P_{i_1}, \ldots, P_{i_k}, SP_{j_1}, \ldots, SP_{j_l} >\} \in TQ$ **Faire**
  $RP_x := \{P_{i_1}, \ldots, P_{i_k}, SP_{j_1}, \ldots, SP_{j_l}\}$
  $QEP_x := \{\}$
  Pour tout $O_y \in RP_x$ **Faire**
    $QEP_x := QEP_x \cup Q_i@O_y$
  Fin Pour
  $QEP := QEP \bowtie_x QEP_x$
Fin Pour

requête final QEP est obtenu, à la fin de la traversée de la requête, en faisant la jointure de tous les sous-plans QEP_x correspondant à chaque sous-requête Q_x. De plus chaque opérateur est annoté avec le (super-)pair dans lequel il sera exécuté.

4.5.2.3 *Illustration*

Le plan d'exécution de la requête Q du pair SAED est illustré dans la figure 4.6. Dans cet exemple, le super-pair *Agriculture* reçoit une requête du pair SAED. Après l'algorithme de routage sémantique pour la sélection des pairs pertinents, la requête annotée générée est envoyée à l'algorithme de mise sous forme algébrique algébrique. Le premier sous-plan généré QEP_1 est pour la requête $Q1$ (pair ISRA est le seul à pouvoir répondre à $Q1$). Le second plan QEP_2 est pour la requête $Q2$. Il y a deux nœuds prometteurs pour $Q2$: super-pair *Pédologie* et pair ISRA. Par conséquent, les requêtes correspondantes $Q2@ISRA$ et $Q2@Pedologie$ sont envoyées aux nœuds correspondants. Le pair ISRA exécute la requête sur ses données et le super-pair *Pédologie* applique le processus de traitement de requête dans sa communauté. Les réponses renvoyées sont réunies plus tard. Finalement le super-pair *Agriculture* joint les résultats des deux sous-plans et les renvoie au pair SAED. L'algorithme de mise sous forme algébrique favorise des résultats de requêtes valides et complets. En effet, en joignant les résultats de tous les fragments de requêtes, nous assurons que le résultat sera valide. D'autre part, la réunion de toutes les réponses possibles pour chaque requête rend possible l'obtention de plus de résultats correctes.

Pour échanger les plans entre les (super-)pairs, SenPeer utilise les canaux de communication introduits dans le chapitre 3. En instanciant un plan, le super-pair initie le processus de création des canaux bidirectionnels nécessaires entre lui et les autres (super-)pairs participant au plan de la requête, comme indiqué dans le chapitre 3. Ces canaux vont servir de conduits à travers lesquels vont transiter les résultats des requêtes, des informations sur les fautes éventuelles et utiles pour l'adaptabilité du plan durant l'exécution de la requête. Notons que les canaux sont asynchrones, ce qui signifie que les données peuvent

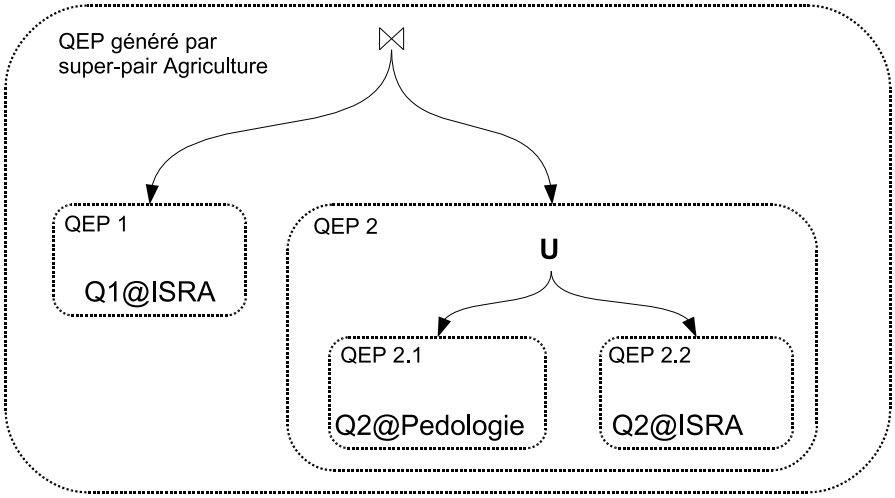


Figure 4.6 – Plan d'exécution pour la requête Q du pair SAED.

être envoyées ou reçues à tout moment, ce qui permet aux (super-)pairs d'agir indépendamment les uns des autres.

4.6 Optimisation de requêtes

Dans ce paragraphe, nous introduisons les détails du processus d'optimisation de requêtes au sein de chaque super-pair. Le but final de l'optimisation d'une requête est de réduire le coût du traitement de la requête en termes de coût de transmission des données mais aussi en termes de coût de traitement des opérateurs de la requête sur les résultats intermédiaires. Ceci est réalisé en reformulant un plan de requête en un autre qui utilise moins de temps et/ou moins de ressources durant l'exécution de la requête. Étant donné un plan de requête algébrique, le but du module d'optimisation de requêtes est de produire une requête optimisée en accord avec un modèle de coût.

4.6.1 Module d'optimisation de requêtes d'un (super-)pair

Chaque super-pair abrite un module d'optimisation de requêtes permettant la génération dynamique de bons plans de requêtes à partir des requêtes qu'il reçoit. En effet, étant donné une requête, il y a plusieurs plans d'exécution possibles pour produire des résultats à la requête. Ces plans permettent de répondre à la même requête et sont équivalents en termes de leurs sorties finales. Cependant, ils peuvent différer en accord avec plusieurs paramètres de performance tels que le temps de réponse et l'utilisation des ressources des (super-)pairs. Le rôle du module d'optimisation est de trouver un moyen alternatif d'exécution de la requête qui minimise une fonction de coût incluant les coûts de traitements I/O, CPU, les coûts de communication mais aussi le temps de réponse. Le module d'optimisation doit générer des plans d'exécution de requêtes efficaces et qui sont effectivement exécutables en respect avec les contraintes imposées par les sources de données abritées par les (super-)pairs. Ceci signifie que des plans optimaux peuvent ne pas être considérés s'il ne sont pas exécutables. Le problème de l'optimisation peut être défini de façon abstraite comme suit :

Définition 4.10. (Optimisation de requêtes) Étant donné un plan d'exécution de requête QEP, un espace d'exécution E qui calcule QEP et une fonction de coût C définie sur E, trouver une exécution e appartenant à E_{QEP} (le sous-ensemble de E qui calcule QEP) de coût minimum $\min_{e \in E_{QEP}} C(e)$.

Un module d'optimisation de requêtes peut être capturé par trois dimensions qui sont[85]:
(i) *l'espace d'exécution* qui capture le modèle d'exécution et définit les exécutions alternatives,
(ii) *le modèle de coût* qui prédit le coût d'une exécution et
(iii) *la stratégie de recherche* qui permet l'énumération des plans d'exécution alternatifs et sélectionne le meilleur d'entre eux.
La figure 4.7 décrit l'architecture d'un module d'optimisation de requêtes d'un super-pair.

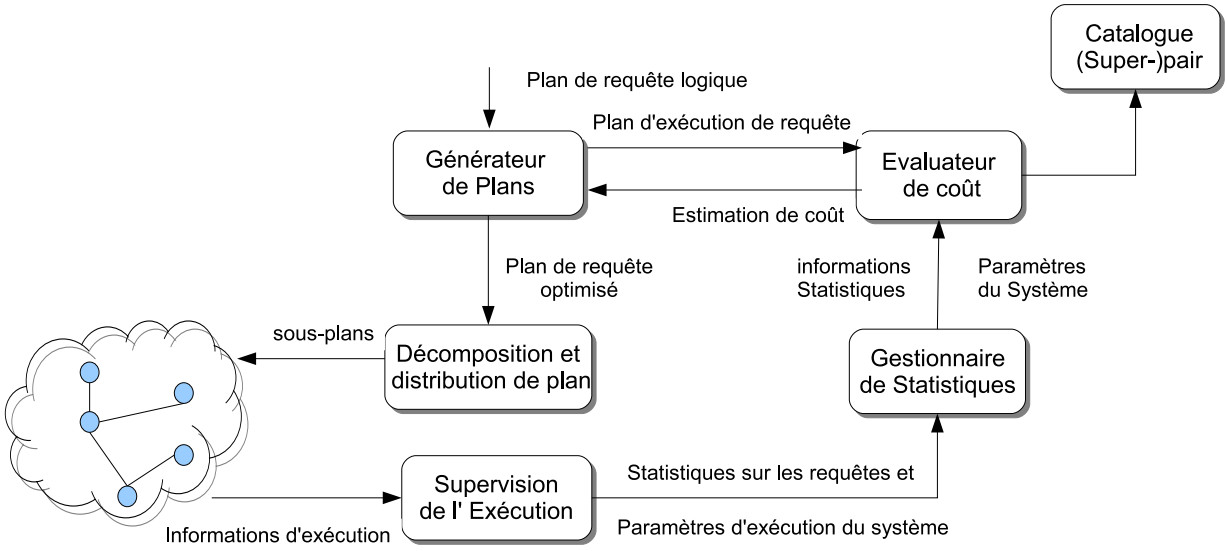


Figure 4.7 – Architecture du module d'optimisation de requêtes d'un (Super-)pair.

A la réception d'un plan de requête en provenance de l'algorithme de mise sous forme algébrique, le générateur de plan applique des règles de transformation et des heuristiques telles que la distribution des *unions* et des *jointures* pour augmenter les performances des plans de la requête. L'étape suivante est le choix du meilleur plan alternatif en utilisant les statistiques obtenues à l'exécution et les paramètres du système incluant la connection des (super-)pairs pertinents, la taille des données stockées à chaque pair, la sélectivité des opérateurs de la requête, en accord avec un modèle de coût prenant en compte des coûts de traitement I/O, CPU et les coûts de communication entre pairs. Une méthode immédiate pour produire un ordonnancement des opérations est de déterminer de façon exhaustive le coût de chacun de ces plans alternatifs et ensuite sélectionner le plan ayant un coût minimum. Finalement, le plan optimisé est envoyé vers le moteur d'exécution de requêtes qui va le décomposer et distribuer les fragments résultant de la décomposition aux (super-)pairs appropriés et va superviser par la suite leurs évaluations.

Après la phase d'optimisation, le parrain super-pair instancie les sous-plans locaux à la communauté tout en distribuant les sous-plans restants vers les super-pairs alliés dans lesquels le processus d'optimisation va continuer.

La variabilité causée par les fautes et les congestions, qui sont des questions inhérentes à l'exécution, affectent l'efficacité du processus de traitement de requêtes parce que les plans d'exécution de requêtes

sont générés de façon statique. Par conséquent, pour ne pas obtenir un résultat non optimal, le plan d'exécution de la requête doit s'adapter à ces changements. Ainsi donc, un processus de traitement de requêtes adaptatif nécessite aussi le changement des canaux correspondants déployés au début de l'exécution de la requête et qui deviennent obsolètes au moment de la faute. Chaque super-pair supervisant l'exécution d'un plan de requête dispose d'un *écouteur* permettant de détecter les fautes à l'exécution. Il est responsable de l'identification de telles fautes, de l'information des nœuds affectés par l'altération du canal de communication et du redéploiement de nouveaux canaux. En cas de faute, l'évaluation de la requête est recommencée en basculant dans un nouveau plan après avoir écarté les résultats partiels précédents.

4.6.2 Equivalences algébriques

Les plans de requêtes sont exprimés de façon naïve par l'algorithme de mise sous forme algébrique des requêtes. Ils sont ensuite assujettis à plusieurs transformations produisant des plans alternatifs et contiennent aussi l'annotation de chaque opérateur virtuel par le site devant le prendre en charge, mais aussi l'ordre dans lequel les opérateurs doivent être invoqués.

A la compilation, la requête initiale est transformée en appliquant des transformations équivalentes des plans de requêtes. En effet, bien que la réécriture algébrique naïve peut générer de bons plans dans certains cas, de meilleurs plans peuvent ne pas être explorés. Nous utilisons des règles de transformation pour explorer les plans de requêtes alternatifs. Les alternatives générées peuvent être triées en utilisant des statistiques telles que le coût de communication et le temps de réponse et la meilleure d'entre elles est exécutée. Ces statistiques peuvent être tirées de l'exécution de requêtes antérieures. Nous précisons que pour minimiser le nombre de messages échangés ainsi que le volume de données transférées nous privilégions les plans de requêtes alternatifs avec moins de sous-plans.

Étant donné que nous appliquons tous les opérateurs de réduction (sélections et projections) avant la transmission des données, le traitement des données concerne principalement le coût des *unions* et des *jointures*. Pour nos règles de réécriture, nous nous focalisons sur les opérateurs de *jointure* et d'*union*, étant donné qu'ils consomment la majorité du temps dans le traitement d'une requête dans un contexte P2P. Nous ne considérons pas les semi-jointures. Bien qu'étant un concept clé du traitement de requêtes distribuées, elles sont spécifiques au modèle relationnel. Étant donné que SenPeer est indépendant des langages de requêtes des pairs, nous avons décidé de les ignorer.

Dans les arbres représentant nos plans de requêtes, les *unions* figurent en bas pour combiner tous les résultats pour chaque sous-requête SQUEL prise en charge par plusieurs pairs. Ces résultats sont combinés plus tard par les jointures placées en haut de l'arbre du plan. Pousser les unions en haut de l'arbre et les jointures près des feuilles (les sources de données) nous permet

(i) d'évaluer une jointure à l'intérieur d'un pair unique et non entre deux pairs distants,
(ii) d'exécuter l'union en parallèle dans plusieurs pairs,
(iii) d'augmenter le degré de distribution et donc de mieux utiliser les capacités de traitement distribuées.

En favorisant le traitement intra-pair (au sein d'un même pair) nous exploitons les bénéfices de l'envoi des requêtes vers les sources plutôt que l'envoi des données vers le pair interrogateur.

Règle 1 : Nous utilisons la règle de transformation suivante[86] qui reformule la jointure d'unions en une union de jointures :

$$R1 \ : \ (\bigcup_{i=1}^{n} T_i) \bowtie (\bigcup_{j=1}^{m} R_j) \Longleftrightarrow \bigcup_{k=1}^{n \times m} (\bowtie_{i,j} \ (T_i, R_j). \tag{4.14}$$

où T_i et R_j sont des sous-plans arbitraires.

Cependant, cette règle de transformation mène à un nombre de sous-plans potentiellement élevé. De ces plans, nous prenons les meilleurs en termes de temps de réponse. Pour la robustesse du traitement de requêtes distribuées, nous favorisons les plans de requêtes avec moins de sous-plans, étant donné qu'il y a moins de messages et de données à transférer.

Règle 2 : Nous rappelons qu'une requête est décomposée en sous-requêtes qui peuvent être éventuellement traitées dans des pairs différents et les résultats joints. Cependant, effectuer les jointures au sein d'un même pair est plus bénéfique parce le coût de la requête est minimisé et les résultats sont renvoyés le plus vite possible. A cet effet, nous identifions l'ensemble des sous-requêtes successives qu'un même pair peut traiter dans le but d'exécuter ces fragments successifs au sein du même pair. Nous utilisons la règle de réécriture suivante.

$$R2 \ : \ \ldots \bowtie Q_1@P_k \bowtie \ldots \bowtie Q_r@P_k \ldots \Longleftrightarrow \ldots (\bowtie_{i=1}^{r} Q_i)@P_k \ldots \tag{4.15}$$

Règle 3 : Pour réduire le volume de données transférées du pair vers le super-pair, qui supervise l'exécution de la requête, les données sont filtrées en utilisant soit un filtre structurel (projection), soit un filtre basé sur le contenu (sélection) ou les deux en même temps sur les sources de données par le biais d'opérateurs de filtrage. Nous transformons la requête en poussant les sélections vers les autres (super-)pairs.

$$R3 \ : \ \sigma(T \ op \ R) \Longleftrightarrow \sigma(T) \ op \ \sigma(R) \ \text{ avec } op \in \{\cup, \bowtie\}. \tag{4.16}$$

En appliquant la règle de transformation $R1$, le plan de la requête QEP dans la figure 4.6 est transformé en un autre plan $QEP3$ dans la figure 4.8. Le plan de requête résultant $QEP3$ 4.6 peut être transformé en appliquant la règle $R2$ en un plan équivalent $QEP4$ comme indiqué dans la figure 4.8. Nous observons que le plan naïf ne prend pas en compte le fait que le pair ISRA peut répondre à plus d'une sous-requête consécutive. En appliquant la règle $R2$ sur le plan QEP3 de la figure 4.8, nous produisons le plan QEP4 dans la même figure. Par conséquent la jointure entre $Q1$ et $Q2$ sera directement exécutée au sein du pair ISRA.

Nous rappelons qu'un opérateur peut être exécuté à un (super-)pair donné si toutes ces entrées sont compilées au sein de ce même (super-)pair. De plus, le choix d'un (super-)pair pour l'exécution d'un opérateur peut être guidé par le coût de communication entre pairs ou les coûts estimés de transfert des résultats intermédiaires. Ce pair doit aussi créer les canaux de communication appropriés pour collecter les résultats intermédiaires.

4.6.3 Énumération des plans alternatifs

Sélectionner un plan optimal est le point focal de notre infrastructure de traitement de requêtes. Bien que l'idée de l'exploration des plans alternatifs assure de trouver le meilleur plan en accord avec un modèle de coût spécifique, elle constitue une tâche difficile exacerbée par le nombre important de (super-)pairs potentiels pouvant participer à la requête. En effet, le module d'optimisation a besoin d'explorer différents plans d'exécution utilisant différents super-pairs pour répondre à une même requête, rendant ainsi l'espace de recherche très vaste. Bien que répondant à la même requête, ces plans peuvent différer, de l'ordre de plusieurs degrés de magnitude, au niveau de leurs coûts. Par conséquent, il est nécessaire de définir des techniques appropriées pour sélectionner le meilleur d'entre eux. Ceci nécessite, d'une part, de définir un modèle de coût pour comparer les différents plans résolvant la même requête et d'autre part de mettre en place une stratégie de recherche de ces plans se basant sur ce modèle de coût.

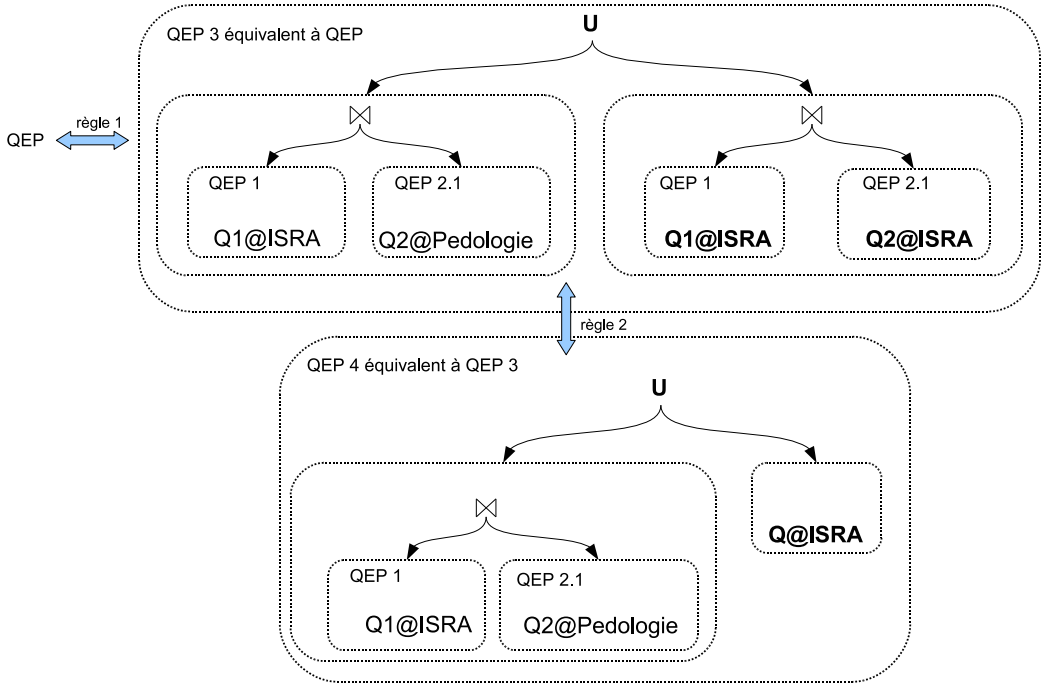


Figure 4.8 – Plan de requête équivalent au plan QEP de la figure 4.6.

4.6.3.1 Stratégies de recherche

Sans perte de généralité, nous supposons que la requête contient N opérateurs virtuels et que chaque opérateur peut être annoté avec M sites (nœuds (super-)pairs). La stratégie d'optimisation va consister aux actions suivantes:

- sélectionner pour chaque opérateur virtuel un site parmi M sites potentiels, ce qui en fait un opérateur concret ;
- ordonner les opérateurs de telle sorte que le plan soit optimal et faisable.

Dès lors, nous sommes confrontés à un problème d'optimisation dans lequel il y a un très grand nombre de solutions (de l'ordre de $O(M^N)$). L'objectif est de sélectionner le plan optimal ou un plan proche de l'optimal dans une échelle de temps réaliste.

Une stratégie presque exhaustive est la programmation dynamique. La programmation dynamique trouve les meilleurs plans pour un petit nombre de relations. Cependant, cette stratégie conduit à un coût de calcul élevé quand le nombre de relations devient élevé, mais aussi elle dépend de statistiques précises qu'il est impossible de maintenir dans un environnement P2P que les pairs peuvent joindre ou quitter à tout moment et en toute liberté. Par conséquent, la programmation dynamique et ses variantes ne sont pas adaptées à la situation d'un PDMS.

Pour éviter le coût élevé de la recherche exhaustive, d'autres manières de déterminer les plans alternatifs, telles que la stratégie de recherche aléatoire, sont utilisées. Ces stratégies essaient de trouver une bonne solution, pas nécessairement la meilleure, mais évitent le coût élevé de l'optimisation. Dans notre processus d'optimisation, nous utilisons les résultats issus de la comparaison de différentes heuristiques d'ordonnancement de jointures et présentés dans[105]. En nous inspirant de ce résumé, nous choisissons l'algorithme de recherche aléatoire appelé 2PO [50](*two-phase optimization*) avec deux phases d'optimi-

sation composées d'une application de l'algorithme d'Amélioration Itérative (*Iterative Improvement*-II) suivie par l'algorithme de Recuit Simulé (*Simulated Annealing*-SA) [51]. En effet, il a été prouvé de façon expérimentale que pour un nombre de relations concernées relativement réduit, de tels algorithmes produisent pratiquement des plans optimaux avec une performance substantiellement améliorée.

La phase d'Amélioration Itérative nous permet de générer une solution initiale dans laquelle la meilleure annotation pour chaque opérateur est choisie localement en respect avec le modèle de coût. Cette solution devient le point de départ de la phase de Recuit Simulé dans laquelle la solution courante (en commençant par la solution initiale) est « perturbée » en changeant les opérateurs virtuels par un ou plusieurs opérateurs annotés et moins coûteux.

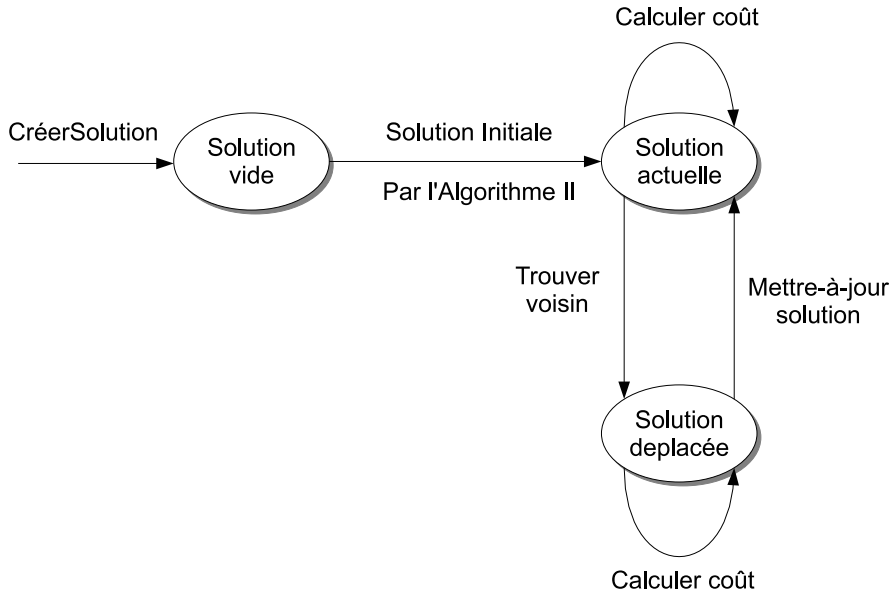


Figure 4.9 – Diagramme d'état du processus d'optimisation à deux phases.

4.6.3.2 L'Amélioration Itérative

L'Amélioration Itérative est une heuristique simple qui améliore la fonction de coût. Elle génère plusieurs solutions initiales qui seront prises comme point de départ de l'exploration de l'espace des solutions. C'est une méthode d'exploration locale qui cherche à améliorer pas à pas la configuration courante. La recherche est effectuée en appliquant une série de déplacements à partir d'un ensemble prédéfini. La fonction de coût est évaluée pour chacun de ces déplacements et elle est maximisée ou minimisée selon le déplacement en se remémorant les solutions obtenues aux étapes précédentes. L'idée principale est de descendre rapidement vers plusieurs minima locaux. Cependant, cette méthode est mise en défaut dans certains cas car ne pouvant pas progresser au delà du premier optimum local rencontré. Il faudrait donc accepter de détériorer provisoirement la configuration courante, pour pouvoir s'échapper des optima locaux et aller explorer d'autres zones de l'espace de recherche.

Algorithme 5 Amélioration Itérative

Sortie : $bestSolution$
$minCout := \infty$
1: **Repeter**
2: $solution :=$"*Solution de départ aléatoire*"
$cout := COUT(solution)$
3: **Repeter**
4: $nouvSolution :=$"*Solution après déplacement aléatoire*"
$nouvCout := COUT(nouvSolution)$
5: **Si** $nouvCout < cout$ **Alors**
6: $solution := nouvSolution$
$cout := nouvCout$
7: **Fin Si**
8: **Jusque** "*Minimum local atteint*"
9: **Si** $cout < minCout$ **Alors**
10: $bestSolution := solution$
$minCout := cout$
11: **Fin Si**
12: **Jusque** "*Temps limite épuisé*"

4.6.3.3 Recuit Simulé

Le Recuit Simulé est une méthode d'optimisation flexible pour les grands problèmes d'optimisation combinatoire. SA est adapté aux problèmes d'optimisation caractérisés par un espace de solutions discret et vaste sur lequel une fonction de coût doit être maximisée ou minimisée (ce qui est trop large pour une recherche exhaustive). Cette méthode permet l'obtention d'une « bonne » solution en une courte période de temps plutôt que de passer plus de temps pour obtenir une meilleure solution .

Le Recuit Simulé (SA) est issu d'une analogie avec le phénomène thermodynamique de recuit des métaux. Le Recuit simulé est appliqué à la suite de la phase d'Amélioration Itérative. Cette phase permet d'explorer de façon plus complète les voisinages des solutions générées par la phase antérieure, en espérant réduire le coût.

Le principe de SA est décrit dans l'algorithme 6. L'algorithme part d'une solution initiale obtenue à partir de la phase II. Dans cette approche, nous supposons que les requêtes sont exprimées en termes des opérateurs et que ces opérateurs ont déjà été annotés avec les (super-)pairs adéquats. De plus, étant donné un opérateur, nous supposons que l'ensemble des (super-)pairs potentiels pouvant l'annoter est déjà connu. Le nombre d'annotations n'est pas le même pour tous les opérateurs de la requête. De façon similaire à II, SA effectue des déplacements aléatoires à partir de la solution initiale en acceptant, avec une certaine probabilité, des déplacements qui résultent en une dégradation de la fonction de coût, ceci pour sortir d'un minimum local. La fonction de coût est celle définie dans le paragraphe 4.6.4. A chaque itération, il tire au sort une modification de la configuration actuelle qui change le coût d'une quantité $\Delta(C)$. Si le coût diminue ($\Delta(C) < 0$), le changement est effectué. Si le coût augmente ($\Delta(C) > 0$), le changement est effectué avec une probabilité $e^{\frac{-\Delta(C)}{T}}$. Le processus se poursuit tant que le coût diminue. Lorsque coût reste stationnaire, la température est diminuée un peu et le processus de décroissance du coût redémarre. Ce processus s'arrête lorsque les diminutions de température restent inefficaces. Il alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser le coût. Ainsi, une nouvelle configuration de coût supérieure à celui de la configuration courante ne sera pas forcément rejetée, son acceptation sera déterminée de façon aléatoire en tenant compte de la température et de la

différence entre les coûts. Le paramètre de température modélise le fait que plus le processus d'optimisation est avancé, moins on est prêt à accepter une solution trop coûteuse, alors qu'au début de la recherche, l'acceptation des solutions fortement coûteuses permettait de mieux explorer l'espace de recherche.

En ce qui concerne la perturbation de la solution ($CREE(Solution)$), nous avons besoin d'identifier la façon la plus appropriée de la modifier tout en la rendant faisable. Pour le type de perturbation à appliquer, nous avons besoin de caractériser la structure du voisinage d'un plan d'exécution donné. Dans l'esprit du recuit simulé, les solutions voisines doivent être similaires à celle actuelle dans le sens qu'elles représentent une légère perturbation de la solution. Par conséquent, nous définissons le voisinage d'une solution comme l'ensemble contenant toutes les solutions qui diffèrent d'elle par la suppression d'un nœud. Cela signifie que nous enlevons un nœud, choisi au hasard, de la solution courante à chaque exécution. Nous continuons à enlever des nœuds de la solution jusqu'à ce que la limite de l'exécution soit atteinte ou jusqu'à ce que nous atteignions une solution infaisable. En procédant ainsi, à chaque fois, la nouvelle solution peut créer un ensemble de nœuds qui peuvent répondre à la requête avec un coût inférieur à celui précédent.

Un objectif important dans notre processus de traitement de requêtes est la découverte du plan optimal ou d'un plan qui lui est proche en respect avec notre modèle de coût en un temps acceptable. A cause de la nature de l'environnement du PDMS, le nombre de solutions à considérer est exponentiel. Ce temps peut être approché en examinant la complexité de l'algorithme. Les paramètres pour le calcul de la complexité sont les suivants :

- N nombre d'opérateurs ;
- M nombre d'annotations possibles par opérateur ;
- T_i température initiale du recuit simulé ;
- T_f Température finale du recuit simulé ;
- τ taux de réduction (refroidissement) de la température.

La phase de recherche de la solution initiale nécessite d'itérer sur tous les N opérateurs virtuels. Pour chaque opérateur, elle a besoin de trouver la meilleure annotation parmi M possibilités. Trouver la meilleure annotation nécessite un tri qui prend $O(M \times \log M)$. Cette phase prend alors $O(N \times M \times \log M)$. La phase suivante dépend de plusieurs paramètres inhérents au processus recuit. La boucle interne tourne pour I itérations. En se basant sur la condition de la boucle externe ($T_i > T_f$) et la façon dont T_i converge ver T_f, le nombre d'itérations pour la boucle externe est approximée par $\log(T_f/T_i)/\log\tau$. En supposant que la fonction de perturbation $CREE(solution)$ a une valeur constante pour son temps d'exécution, la complexité de l'algorithme est alors : $O(N \times M \times \log M + I \times \log(T_f/T_i)/\log\tau)$.

Ainsi donc le temps de génération des plans dépend non seulement du nombre d'opérateurs qui doivent être réordonnés, mais aussi du nombre de (super-)pairs candidats pour exécuter les opérateurs de la requête. D'autre part, l'algorithme met plus de temps pour trouver une solution optimale. Ceci est prévisible car le recuit simulé essaie d'attendre un état d'équilibre à travers plusieurs répétitions. Cependant, le succès et la qualité du recuit simulé reposent sur le choix de paramètres adéquats. Certains d'entre eux dépendent du problème, d'autres sont spécifiques à l'implémentation.

4.6.4 Modèle de coût

Les systèmes P2P sont très dynamiques au regard de l'ensemble des pairs connectés, du nombre de sources de données disponibles, de la charge de travail et de la consommation des ressources au niveau des (super-)pairs. Par conséquent, notre modèle de coût pour le traitement des requêtes dans notre PDMS doit prendre en compte ces paramètres. Certains de ces paramètres sont détenus par les super-pairs dans leurs catalogues, tandis que d'autres, comme la situation de la charge, sont déterminés périodiquement

Algorithme 6 Recuit simulé

Entrées : Solution initiale obtenue par la phase II
Température initiale T_i
Température finale T_f
Taux de diminution de la température τ
Nombre maximum d'itérations max_I
Sortie : Plan optimal

Pour tout $Vop \in op(QEP)$ **Faire**
  $solution = solution \cup \{MeilleurAnnotation(Vop)\}$//solution initiale obtenue par la phase II
Fin Pour
$bestSolution := solution$
$T := T_i$
$cout := COUT(bestSolution)$
$minCout := cout$
TantQue $(T_f < T)$ **Faire**
  Pour $I := 1$ *a* max_I **Faire**
    $nouvSolution := CREE(solution)$// Choix au hasard d'un voisin de $solution$
    $nouvCout := COUT(nouvSolution)$
    $\Delta := nouvCout - cout$
    Si $\Delta \leq 0$ **Alors**
      $solution := nouvSolution$
      $cout := nouvCout$
    Sinon
      Si $(e^{\frac{-\Delta}{T}} \geq RAND(0..1))$ **Alors**
        $solution := nouvSolution$
        $cout := nouvCout$
      Fin Si
    Fin Si
    Si $cout < minCout$ **Alors**
      $bestSolution := solution$
      $minCout := cout$
    Fin Si
  Fin Pour
  $T := T * \tau$
Fin TantQue

par l'environnement d'exécution.

Dans le but de mettre en œuvre une bonne stratégie de traitement de requêtes, nous devons être en mesure de calculer le coût d'exécution de chaque plan de requête et de le comparer aux coûts des plans alternatifs. A cet effet, nous introduisons les paramètres du modèle de coût que nous utilisons dans le but d'évaluer le coût de traitement des plans de requêtes alternatifs dans notre PDMS. Il existe déjà, dans la littérature, des travaux assez vastes à propos des modèles de coût pour différents opérateurs et différentes configurations de systèmes. Notre modèle de coût doit être en mesure d'estimer la consommation totale de ressources et le temps de réponse d'un plan de requête donné.

4.6.4.1 Fonction de coût

Le coût total est la somme de tous les temps passés à traiter les opérateurs de la requête dans différents (super-)pairs et à transférer les résultats intermédiaires entre les nœuds participant à la requête tandis que

le temps de réponse est le temps passé à exécuter entièrement la requête. Pour ce faire, nous utilisons des fonctions évaluant le coût de chaque opérateur en se basant sur des statistiques et des formules permettant d'estimer la taille des résultats intermédiaires et la sélectivité des opérateurs de la requête. Nous favorisons les plans de requête minimisant le coût total et le temps de réponse rendant ainsi possible la prise en compte du parallélisme. Minimiser le temps de réponse, c'est-à-dire, exploiter toutes les exécutions parallèles possibles, n'implique pas la minimisation du coût total. Bien au contraire, le coût total pourrait augmenter à cause de l'augmentation du traitement effectué en parallèle. Ceci pourrait être indésirable dans le cas des système P2P et un compromis entre ces deux métriques doit être fait. Pour estimer le traitement intra et inter-pair, nous utilisons une estimation de coût qui est une somme pondérée du coût total et du temps de réponse. Le coût est pondéré dans le but de modéliser l'impact des machines lentes et de celles rapides, mais aussi l'effet de l'occupation des canaux de communication. La formule correspondante est indiquée ci-dessous :

$$cout = \omega_s S + \omega_p P + \omega_r R. \tag{4.17}$$

où la dernière composante R représente le temps de réponse et les deux premières (S-*coût de l'envoi* et P-*coût de traitement*) le coût total. Les poids associés sont ω_r pour le temps de réponse, ω_s pour l'envoi et ω_p pour le traitement.

L'équation 4.17 ne prend pas en compte les facteurs tels que la charge du réseau et les caractéristiques dynamiques des (super-)pairs. Il est important de connaître la charge de travail du super-pair et de ses voisins (super-)pairs pour éviter toute surcharge quand trop de requêtes exécutent des opérateurs au niveau des super-pairs. Ces facteurs sont cruciaux pour l'optimisation de requêtes distribuées dans un PDMS et donc, nous ne pouvons pas les ignorer. Par conséquent, le modèle de coût doit être raffiné pour les prendre en compte dans le choix de meilleurs plans. Nous introduisons donc les paramètres L_{SP} et L_N qui sont respectivement la charge de travail du (super-)pair (par exemple 30%) et la charge du réseau. Le coût de traitement est alors divisé par une quantité $(1-L_{SP})$ liée à la charge de travail du (super-)pair. Le coût de communication est ajusté de la même façon en le divisant par une quantité $(1 - L_N)$ liée à l'utilisation actuelle de la bande passante vers le (super-)pair devant participer au plan. Notre nouvelle estimation de coût devient alors :

$$Cout(QEP) = \omega_s \frac{S(QEP)}{1 - L_N} + \omega_p \frac{P(QEP)}{1 - L_{SP}} + \omega_r R(QEP). \tag{4.18}$$

Nous utilisons donc la connaissance sur la charge comme un paramètre important du modèle de coût. Les connaissances sur les charges sont échangées périodiquement entre les (super-)pairs liés. De Cette façon, le module d'optimisation dispose d'une vue sur la charge qui lui permette de décider si un sous-plan doit être envoyé à un allié super-pair ou plutôt s'il doit explorer un plan de requête alternatif qui serait moins coûteux.

4.6.4.2 Temps de réponse

Si tous les opérateurs du plan sont exécutés de façon séquentielle, le temps de réponse d'une requête est égal à son coût. Cependant, si le parallélisme est exploité, le temps de réponse d'une requête peut être plus faible que le coût total.

Le temps d'exécution d'un plan de requête parallèle est calculé en termes du temps de réponse des (super-)pairs participant au plan de la requête. Ce temps est proportionnel au nombre de fois que la source est sollicitée mais aussi au nombre de n-uplets à transférer à chaque sollicitation de la source. Soit QEP

un plan de requête avec les sous-plans $\{QEP_1, \ldots, QEP_k\}$, son temps de réponse est fonction du temps de réponse de chacun de ses sous-plans comme suit :

$$R(QEP) = \sum_{i=1}^{k} TempsReponse(QEP_i) \tag{4.19}$$

Le temps de réponse de chaque sous-plan $QEP_i@(P_{i1}, \ldots, P_{im})$ est calculé en termes de chaque (super)-pair individuel participant au sous-plan. Étant donné que les sources sont traitées en parallèle, le temps de réponse cumulatif est défini comme le maximum des temps de réponse de toutes les sources participant au sous-plan.

$$TempsReponse(QEP_i) = \max_{1 \leq j \leq m} \{TempsReponse(QEP_i@P_{ij})\} + \alpha \sum_{j \neq \max_{TR}} TempsReponse(QEP_i@P_{ij}). \tag{4.20}$$

avec :

- $\max_{TR} = \arg\max_{1 \leq j \leq m} \{TempsReponse(QEP_i@P_{ij})\}$;
- $\alpha \in [0, 1]$ indiquant le degré de parallélisme supporté par le système (0 pour un parallélisme total et 1 pour une exécution strictement séquentielle).

Puisque nous avons supposé que le système supporte un parallélisme total, le dernier terme de l'équation 4.20 s'annule.

Pour l'évaluation du temps de réponse au sein de chaque source, nous utilisons la métrique de temps de réponse qui a été étudiée dans [86] et représentée par la formule ci-dessous :

$$r = C_{CPU} * \alpha(inst) + C_{I/O} * \alpha(I/O) + C_{MSG} * \alpha(msg) + C_{TR} * \alpha(paquet). \tag{4.21}$$

où C_{CPU}, $C_{I/O}$, C_{MSG} et C_{TR} indiquent respectivement les opérations suivantes : instruction CPU, instruction d'entrée/sortie (I/O) disque, création d'un message et transmission d'un message. $\alpha(o)$ est le nombre maximum de o opérations qui doivent être faites de façon séquentielle pour exécuter la requête. Les deux premières composantes évaluent le temps de traitement tandis que les deux dernières calculent le temps de communication. Cette formule considère l'exécution parallèle étant donné qu'aucun traitement et communication fait en parallèle n'est pas pris en compte. Le temps de réponse total du plan de requête est donné en sommant le coût de traitement et de communication de chaque opérateur.

4.6.4.3 Coût de l'envoi

Le coût de l'envoi $SCout$ est le coût du transfert des données d'un nœud à un autre. Ce coût est évalué en tenant en compte la taille des données échangées et les propriétés physiques du réseau e.g, latence $latence$, et bande passante $bandePassante$. La formule est donnée ci-dessous :

$$SCout(Q) = latence + |Q|/bandePassante. \tag{4.22}$$

Dans les bases de données distribuées, le coût de l'envoi est souvent plus grand que le coût de traitement et influence largement le coût total. Étant donné que dans notre PDMS nous favorisons le plus possible le traitement au sein d'un même pair en poussant les jointures vers les sources de données, la taille des résultats intermédiaires est alors plus petite, le coût d'envoi aussi. Dans ce cas, le coût de traitement pourrait être plus significatif étant donné aussi les progrès dans les connections des pairs qui sont de plus en plus rapides.

4.6.4.4 Coût de traitement

Le coût de traitement total d'un plan de requête est la somme des coûts de traitement de chaque opérateur dans l'arbre de la requête. Ce coût dépend du temps CPU de chacun de ces opérateurs qui dépendent, à leur tour, des tailles des relations qu'ils prennent en entrée. Dans la suite, nous considérons les opérateurs de plan qui consomment le plus de temps, la jointure et l'union. Avant de donner les formules concernant les coûts des opérateurs, nous introduisons celles permettant de calculer la taille des résultats intermédiaires, ou encore leur cardinalité.

La cardinalité de la jointure est le nombre de lignes produites lorsqu'on effectue la jointure de deux ensembles de lignes. Une jointure est le produit cartésien de deux ensembles de lignes, avec le prédicat de la jointure appliqué comme filtre sur le résultat. Par conséquent, la cardinalité de la jointure est estimée par le produit des cardinalités des deux ensembles de lignes, multiplié par la sélectivité du prédicat de jointure. La cardinalité de la jointure est estimée comme suit :

$$|R \bowtie T| = |R| * |T| * selectiviteJoin. \tag{4.23}$$

où $|R|$ représente le nombre de tuples de R. La sélectivité de la jointure est une valeur entre 0 et 1 définissant un rapport entre les tuples retenus par la jointure et ceux créés par le produit cartésien.

$$selectiviteJoin = |R \bowtie T|/|R \times T|. \tag{4.24}$$

Une jointure est dite avoir une meilleure sélectivité s'il a un facteur de sélectivité plus faible. Nous pouvons remarquer que dans l'équation 4.24 , il est impossible de déterminer la sélectivité de la jointure avant la jointure elle même, et donc avant l'évaluation de la requête. Dans notre contexte nous supposons que la sélectivité de chaque jointure est estimée et disponible dans le catalogue du (super-)pair. Nous pouvons par exemple utiliser une borne supérieure($|R \times T|) = |R| * |T|$) divisée par une constante pour refléter le fait que le résultat de la jointure est plus petit celui du produit cartésien. Après l'exécution de chaque requête, l'estimation initiale de chaque jointure est réévaluée et elle est rendue plus réaliste.

Comme suggéré dans [40], la cardinalité de l'opérateur d'union peut être estimée comme suit :

$$|\bigcup_{i}^{n} R_i| = \sum_{i=1}^{n} |R_i|/n + \max_i(|R_i|). \tag{4.25}$$

Intuitivement, ceci est dû au fait que la cardinalité d'une union ne sera jamais plus petite que la cardinalité du plus grand ensemble participant à la réunion ($\geq \max_i(|R_i|)$ et elle est aussi grande que la somme des tailles des opérandes ($\leq \sum_{i=1}^{n} |R_i|$). Par conséquent, une valeur proche du milieu semble être le choix le plus approprié, c'est-à-dire la moyenne de la somme ajoutée au plus grand.

En utilisant les formules présentées ci-dessus, nous pouvons maintenant introduire le coût de traitement des opérateurs de jointure et de réunion :

$$\begin{cases} PCost(R \bowtie S) = Jinit + |R \bowtie S|/joinRatio \\ UCost(R \cup T) = Uinit + |R \cup T|/unionRatio \end{cases} \tag{4.26}$$

$Jinit$ ($Unit$) est une constante de temps indiquant le temp nécessaire pour générer l'opération de jointure(union), démarrer la transmission et le contrôle de sécurité. Ce rapport est utilisé pour simuler la vitesse du processeur du (super-)pair.

Notons que les formules ci-dessus ne prennent pas en compte la taille de chaque tuple, mais considèrent seulement les cardinalités des relations. En effet, étant donné la nature d'un PDMS comme SenPeer, il semble surréaliste de maintenir un modèle de coût plus raffiné, étant donné que l'on ne peut

pas disposer de statistiques précises dans un environnement P2P, ni exercer un contrôle sur le temps de traitement d'un pair distant.

4.6.5 Décomposition et distribution des plans de requêtes

Une fois le plan optimal déterminé, le super-pair initie une phase de distribution des différents sous-plans vers les différents (super-)pairs impliqués dans le traitement de la requête. Rappelons que les plans de requêtes doivent être complètement annotés (les sous-plans avec les nœuds pouvant les prendre en charge les opérateurs avec les nœuds sur lesquels ils sont exécutés) pour qu'ils puissent être valides.

Le meilleur plan de requête produit par le module d'optimisation est divisé en une partie locale, destinée à être exécutée dans la communauté dans laquelle la requête a été initiée, et en une partie distante contenant un ensemble de sous-plans distants destinés à être exécutés dans les communautés alliées contenant les super-pairs participant à la requête. Le plan local est alors instancié et les plans distants envoyés vers les communautés alliées. Nous rappelons que ces super-pairs vont continuer le processus de routage et d'optimisation sur des plans plus petits. L'algorithme 7 montre comment le plan de requête est divisé en utilisant un parcours en profondeur de l'arbre de la requête.

Algorithme 7 Décomposition et distribution des sous-plans

$fragmentQEP(QEP, op)$
$Marquer(op)$
Pour tout $opFils \in Fils(op)$ **Faire**
  Si $nonMarque(op)$ **Alors**
    Si $Site(opFils) = Node(op)$ **Alors**
      $fragmentQEP(QEP, opFils)$
    Sinon
      $ajouter(QEP_{Rem}, Site(opFils), envoiPlan(Site(op), opFils)$
      $remplacer(QEP_{Loc}, opFils, receptResult(Site(opFils))$
    Fin Si
  Fin Si
Fin Pour

Nous utilisons les notations suivantes :

- QEP_{Loc} et QEP_{Rem} indiquent respectivement le plan local et le plan distant. QEP_{Rem} spécifie les sites distants sur lesquels exécuter les sous-plans restants ;
- $Marque(op)$ opère un marquage d'un opérateur, ici un nœud de l'arbre pour indiquer s'il a été visité ou pas par le parcours en profondeur ;
- $Fils(op)$ indique l'ensemble des nœuds fils de l'opérateur op ;
- $Site(op)$ renvoie le site sur lequel est exécuté l'opérateur op ;
- $envoiPlan(Site(op), opFils)$: permet de pousser le sous-plan ayant pour racine $opFils$ depuis le site local $Site(op)$ vers un site distant, c'est-à-dire $Site(opFils)$
- $receptResult(Site(opFils))$ permet la réception les données de l'exécution du sous plan distant de racine $opfils$ après son exécution sur le site distant $Site(opFils)$

La fonction est appelée récursivement avec l'opérateur racine op de l'arbre du plan de la requête et les nœuds de l'arbre de la requête sont aussi visités. Si un opérateur fils $opFils$ est exécuté sur le même site que son parent op, c'est-à-dire, le site local, la fonction est appelée récursivement. Dans le cas contraire, nous venons de découvrir la racine d'un sous-plan distant. Dans ce cas, la fonction $envoiPlan(Site(op), opFils)$ permet de pousser le sous-plan incluant l'opérateur fils $opFils$ du site

local $Site(op)$ vers le site distant, c'est-à-dire $Site(opFils)$. Le sous-plan distant est donc séparé du sous-plan local. L'opération $receptResult(Site(opFils))$ permettra d'ajouter les données reçues depuis le site distant à l'emplacement spécifié lors de la fragmentation. Elle permet donc d'interconnecter le plan local avec les plans distants. Par ailleurs, le processus d'optimisation continue sur ces plans distants et ceci dans les sites distants sur lesquels ils vont s'exécuter.

Le processus de distribution des sous-plans illustré dans la Figure 4.10. Dans cette figure nous avons omis les connections physiques entre nœuds par souci de lisibilité.

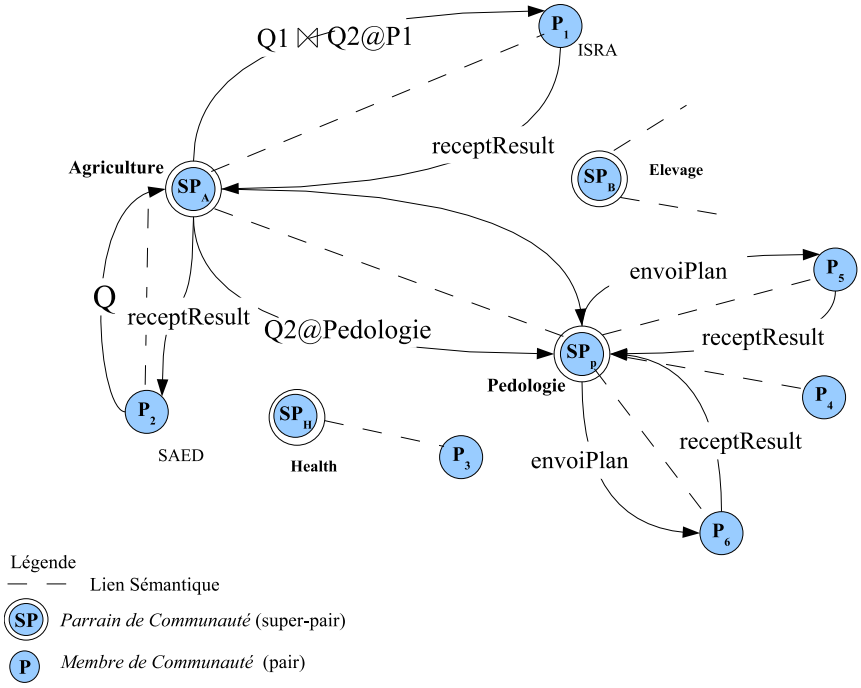


Figure 4.10 – Traitement de requêtes distribuées. Instanciation des plans locaux et distribution des plans distants

4.7 Conclusion

Dans ce chapitre, nous avons introduit le processus de traitement de requêtes dans SenPeer. Le traitement de requêtes est le principal service qu'un pair peut fournir au PDMS. Dans un premier temps, nous avons introduit la sémantique de l'interrogation dans le cadre de notre PDMS. La sélection des pairs pertinents pour une requête repose sur une comparaison entre le sujet de la requête et les expertises connues par le super-pair actuel. Cette mise en correspondance est à la base de l'algorithme de routage sémantique, qui exploite la topologie sémantique induite par la connaissance sur les expertises des pairs, pour annoter chaque requête avec l'ensemble des (super-)pairs pouvant la prendre en charge. Une fois les sites pertinents identifiés, l'algorithme de reformulation permet sur la base des correspondances sémantiques de reformuler une requête entre (super-)pairs sémantiquement liés. Nous avons montré la validité de la reformulation à travers les notions d'inclusion de requêtes et de réponse certaine adaptées à notre contexte. La requête annotée est ensuite passée au générateur de plans qui met en place un plan de

requête distribuée tenant en compte la distribution horizontale et verticale des données. Ce plan naif est optimisé en explorant l'ensemble des plans alternatifs pouvant produire le même résultat et à moindre coût. Nous utilisons un modèle de coût qui dépend des coûts de traitement et d'envoi, mais aussi de la charge de travail au niveau des (super-)pairs à chaque instant, ce qui permet de prendre en compte les changements dynamiques au sein du PDMS.

CHAPITRE 5
Validation

5.1 Introduction

Dans les chapitres précédents, nous avons présenté un ensemble de techniques utilisées dans SenPeer. Dans ce chapitre, nous introduisons les simulations réalisées en vue de l'évaluation de ces techniques. Nous introduisons successivement les validations du processus de découverte de correspondances, du routage sémantique et de l'optimisation de requêtes Pour chaque technique nous présentons le modèle de réseau utilisé pour la simulation ainsi que les métriques de performance utilisées. Nous simulons les techniques pour étudier la performance des différents mécanismes en utilisant différents paramètres.

5.2 Évaluation de la découverte de correspondances

Nous avons abordé le processus d'appariement de schémas d'une façon adaptée à différentes applications et langages de schémas. Nous avons cependant illustré le processus sur des données liées à la mise en valeur de la vallée du fleuve sénégal. Pour évaluer l'efficacité de notre processus de réconciliation sémantique, nous l'avons appliqué à plusieurs situations. Les expérimentations ont été réalisées avec les principaux objectifs suivants :

- déceler l'exactitude du processus de découverte de correspondances sémantiques
- mesurer la contribution de chaque composante d'appariement dans le résultat

5.2.1 Paramètres d'évaluation

5.2.1.1 Schémas sources

Pour une évaluation correcte et fiable du processus de découverte de correspondances, les schémas doivent être variés. Pour ces raisons, les schémas sélectionnés varient en tailles et en caractéristiques. Nous avons évalué le processus de réconciliation sur des données synthétiques (sans instances de données). Elles ont été récoltées depuis le site de l'OMVS [1] (Organisation pour la Mise en Valeur de la vallée du fleuve sénégal), une organisation inter-étatique regroupant les pays bordant le fleuve. Les caractéristiques des schémas utilisés sont indiquées dans la table 5.1.

Les tailles des schémas testés varient entre 83 et 58 nœuds (uniquement les éléments de schémas, les nœuds types et types de données ne sont pas comptabilisés). Le nombre de liens sémantiques liant ces éléments de schémas est compris entre 69 et 92.

Les figures 5.1 et 5.2 illustrent respectivement des parties des schémas suggérés pour les communautés *Hydrologie* et *Activités Hydro-agricoles*. Les schémas sont importés de leur modèle de données d'origine vers notre représentation interne et sérialisés sous format XML en vue de leur échange dans le réseau.

[1] http://www.omvs-soe.org/indexp.htm

Communauté	Schéma	Nombre de nœuds [a]	Nombre d'arcs [b]
Hydrologie	Hydrologie (HY)	67	74
Activités hydro-agricoles	Activités hydro-agricoles (AC-HY)	83	92
	SAED (SA) [c]	64	77
	ADRAO (AD) [d]	58	69

[a] Les nœuds types et types de données ne sont pas comptabilisés.
[b] Arcs liant les de nœuds considérées.
[c] Société d'Aménagement et d'Exploitation des terres du Delta du fleuve sénégal.
[d] Centre du riz pour l'Afrique.

Table 5.1 – Caractéristiques des schémas testés.

Nous avons effectué trois processus de découverte de correspondances impliquant les couples de schémas suivants : HY/AC-HY, AC-HY/SA, et AC-HY/AD. Les schémas sont deux-à-deux différents bien qu'étant dans la même communauté sémantique ou dans des communautés sémantiques liées.

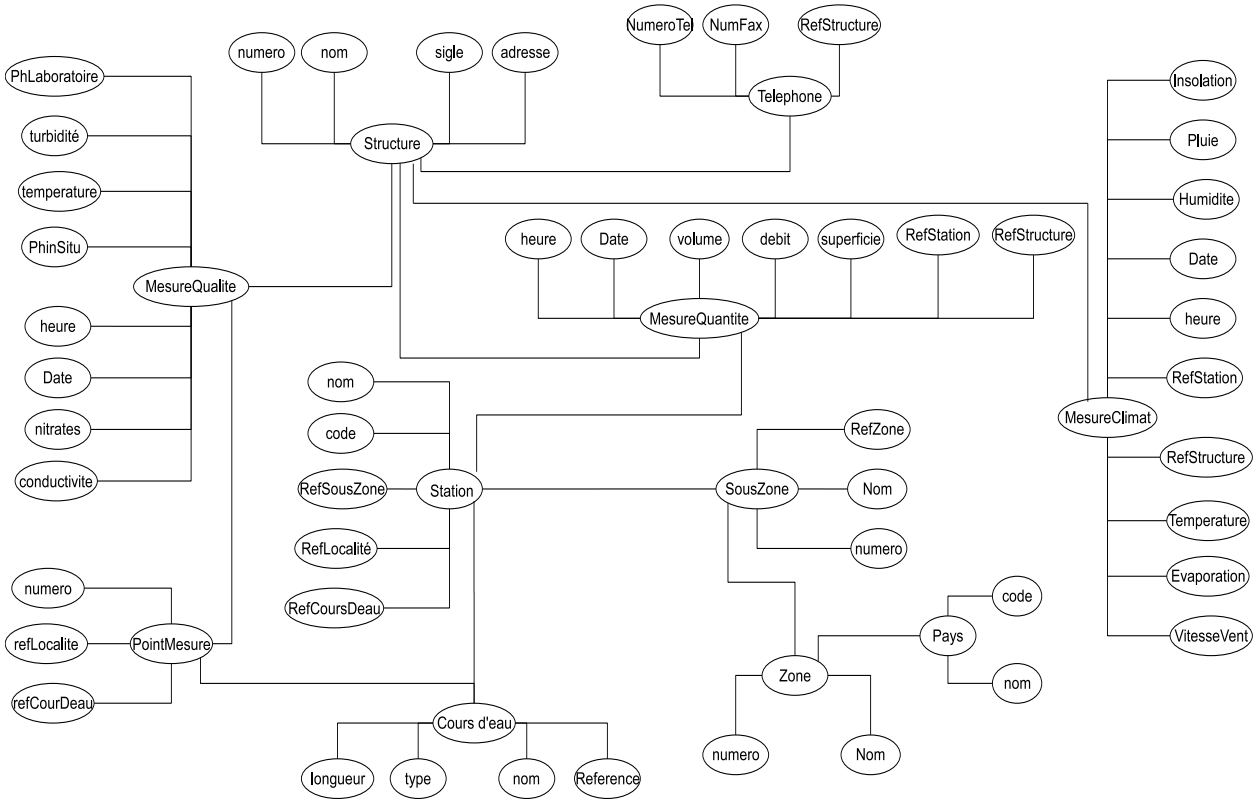


Figure 5.1 – Extrait du $sGraph$ de l'*Hydrologie*

Dans le but de fournir une base pour l'évaluation de la qualité de la découverte automatique de correspondances, nous définissons d'abord, pour chaque couple de schémas, les correspondances réelles de façon manuelle. Ces correspondances réelles peuvent être considérées comme étant celles définies par les experts humains et elles représentent une base pour évaluer la qualité des résultats déterminés automatiquement par le processus de découverte de correspondances sémantiques. Ainsi donc, à partir des

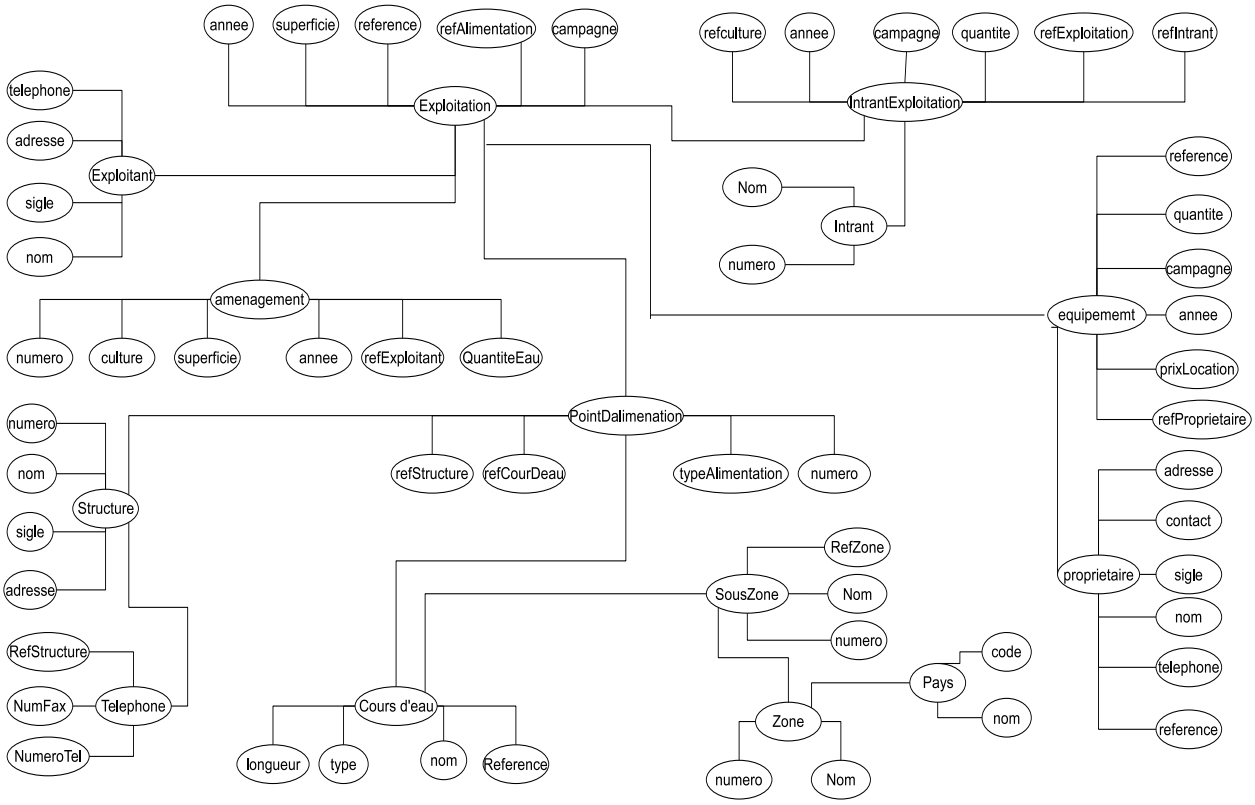


Figure 5.2 – Extrait du $sGraph$ des *Activités Hydro-agricoles*.

correspondances dérivées automatiquement par le système et des correspondances réelles, nous donnons dans le paragraphe suivant les mesures de qualité pour le processus de découverte de correspondances.

5.2.1.2 Méthodologie d'évaluation

Notre objectif est de concevoir un mécanisme efficace de découverte de correspondances sémantiques dans un environnement P2P en présence de plusieurs modèles de données.

Pour une évaluation quantitative de la procédure d'appariement, nous nous basons sur des mesures empruntées au domaine de la Recherche d'Informations, en l'occurrence le *rappel* et la *précision*, et qui permettent d'établir la performance de la recherche. Ces mesures sont calculées sur la base des correspondances produites par le système (la zone à l'intérieur cercle étiqueté par S dans la figure 5.3) et un ensemble complet de correspondances de référence H considérées comme étant correctes (la zone aux contours discontinus). L'ensemble H est souvent établi par les experts humains. Dans la suite, nous faisons référence à l'ensemble de toutes les correspondances possibles (c'est-à-dire le produit cartésien des deux $sGraph$ en entrée) par M. Finalement toutes les correspondances peuvent être classifiées comme suit :

- les correspondances correctes trouvées par le système (appelées *True Positive*) :

$$TP = S \cap H. \tag{5.1}$$

- les correspondances incorrectes retournées par le système (appelées *False Positive*) :

	Prédites Positives	Prédites Négatives
Positives Actuelles	*True Positive* (TP)	*False Negative* (FN)
Negatives Actuelles	*False Positive* (FP)	*True Negative*(TN)

Table 5.2 – Matrice de confusion.

$$FP = S - S \cap H. \tag{5.2}$$

- les correspondances correctes non retournées par le système (appelées *False Negative*) :

$$FN = H - S \cap H. \tag{5.3}$$

- les correspondances incorrectes non retournées par le système (*True Negative*)

$$TN = M - S \cup H. \tag{5.4}$$

Les correspondances dans H sont appelées *correspondances positives* et celles dans $N = M - H = TN + FP$ sont appelées *correspondances négatives*.

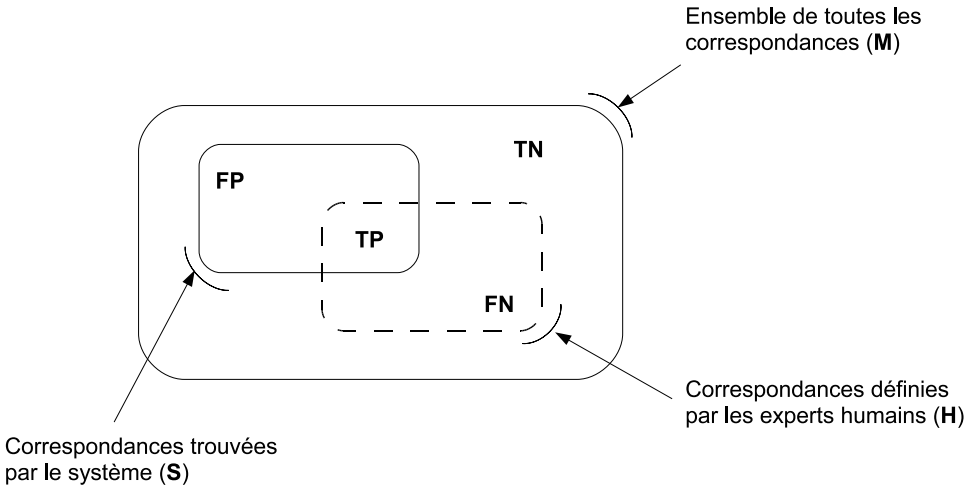


Figure 5.3 – Classification des correspondances sémantiques.

Ces ensembles peuvent être représentés dans une matrice de confusion représentant les relations entre les correspondances actuelles et celles prédites.

A partir de ces ensembles, et en suivant les approches précédentes de découverte de correspondances, nous évaluons la performance de notre approche sur la base des métriques suivantes : le rappel, la précision et leur moyenne harmonique $F\text{-}mesure$[30] :

- La précision est une mesure de validité variant dans l'intervalle $[0, 1]$. Elle indique la fraction de correspondances trouvées qui sont pertinentes :

$$Precision = \frac{|TP|}{|TP + FP|} = \frac{|H \cap S|}{|S|}. \tag{5.5}$$

La précision donne une estimation de l'effort post-appariement à fournir pour supprimer les correspondances *False Positive* renvoyées par le système.

- Le rappel est une mesure de indiquant à quel point les résultats sont complets et elle varie dans l'intervalle $[0, 1]$; Elle indique la fraction des correspondances pertinentes qui ont été trouvées :

$$Rappel = \frac{|TP|}{|TP + FN|} = \frac{|H \cap S|}{|H|}. \tag{5.6}$$

Le rappel donne une estimation de l'effort post-appariement à fournir pour ajouter les correspondances *False negative* non renvoyées par le système.

Intuitivement, les correspondances *False Negative* et celles *False Positive* réduisent la qualité de l'appariement. Dans le cas idéal, si aucune correspondance *False Negative* ou *False Positive* n'a été retournée $Precision$ et $Rappel$ seront égaux à 1.

Le rappel et la précision sont inversement liés. En outre, ni le rappel, ni la précision ne peuvent évaluer séparément et précisément la qualité du processus d'appariement. En particulier, le rappel peut être maximisé au dépens d'une précision pauvre en retournant toutes les correspondances possibles, c'est-à-dire, le produit cartésien des deux $sGraph$. Au même moment, une précision élevée peut être atteinte au dépens d'un rappel pauvre en retournant seulement un petit nombre de correspondances (correctes). Par conséquent, il est nécessaire de disposer d'une mesure globale impliquant à la fois le rappel et la précision et permettant d'évaluer la qualité de l'appariement. Pour cela, nous utilisons la métrique $F\text{-}mesure$ qui établit une moyenne harmonique du rappel et de la précision comme suit[117] :

$$F\text{-}mesure = \frac{2}{\frac{1}{Rappel} + \frac{1}{Precision}} = \frac{2 \times Rappel \times Precision}{Precision + Rappel}. \tag{5.7}$$

Dans l'équation ci-dessus le rappel et la précision sont aussi importantes l'un que l'autre.

Nous utilisons aussi la métrique $Globale$ (*overall*) qui a été spécialement introduite dans le cadre de la découverte de correspondances. Le but de cette métrique est de quantifier les efforts post-appariement déployés pour ajouter les correspondances manquantes ou enlever celles erronées[117].

$$Globale = 1 - \frac{|FN| + |FP|}{|TP| + |FN|} = \frac{|TP| - |FP|}{|FN| + |TP|} = Rappel * (2 - \frac{1}{Precision}). \tag{5.8}$$

La métrique $F\text{-}mesure$ est plus optimiste que la métrique $Globale$. Pour des mêmes valeurs de $Rappel$ et de $Precision$, $F\text{-}mesure$ est toujours plus élevée que $Globale$. A la différence des trois autres mesures, $Globale$ peut avoir des valeurs négatives si le nombre de correspondances classées *False Positive* excède le nombre de correspondances *True Positive* (c'est-à-dire si $Precision < 0.5$)

Tous les résultats ont été obtenus en configurant l'algorithme de découverte de correspondances avec les valeurs des paramètres indiqués par le tableau 5.3.

Paramètre	Description	Valeur
ω_s	poids de la similarité de synonymes	0.6
ω_t	poids de la similarité de types	0.4
λ_l	poids de la similarité linguistique	0.5
λ_v	poids de la similarité de voisinage	0.5
ϵ_{acc}	seuil minimum de similarité	0.6

Table 5.3 – Valeurs des paramètres pour l'appariement.

5.2.2 Résultats

Dans ce paragraphe, nous présentons les qualités des correspondances issues du processus de réconciliation. Tous les algorithmes de découverte de correspondances ont été implémentés en Java et ils ont été évalués sur un Pentium IV avec une fréquence de 2.90 GHz, une mémoire de 512 Mo et un disque dur de 40 GB tournant sous le système Windows XP.

Nous avons évalué l'impact de la combinaison de différents appariements sur la qualité de la réconciliation dans les différentes configurations testées. Pour chaque cas de test, nous effectuons la combinaison de différents appariements. Les mesures de qualité ont d'abord été définies pour chaque expérience et ensuite la moyenne a été faite sur les expériences de chaque série.

La figure 5.4 montre les valeurs moyennes pour les quatre mesures de qualité qui ont été utilisées ($Rappel$, $Precision$, $F-mesure$, et $Globale$) pour chaque appariement ou combinaison d'appariements. Tandis que l'appariement SYN+TYPE atteint une valeur de $Globale$ de 0.59 et de $F-mesure$ de 0.79 ($Precision = 0.8$, $Rappel = 0.78$), la combinaison LING+VOIS atteint une valeur de $Globale$ de 0.72 et de $F-mesure$ de 0.88 ($Precision = 0.86$, $Rappel = 0.88$).

Globalement les combinaisons d'appariement sont de meilleures qualités que les appariements uniques.

Dans la dernière combinaison LING+VOIS, qui est celle réellement utilisée dans SenPeer, les trois premières mesures excèdent toutes la valeur 0.86(la meilleure valeur étant 1) tandis que la valeur de la mesure $globale$ tourne autour de 0.72. La limitation de cette dernière mesure autour de 0.72 est apparemment influencée par le degré de similarité modéré des schémas.

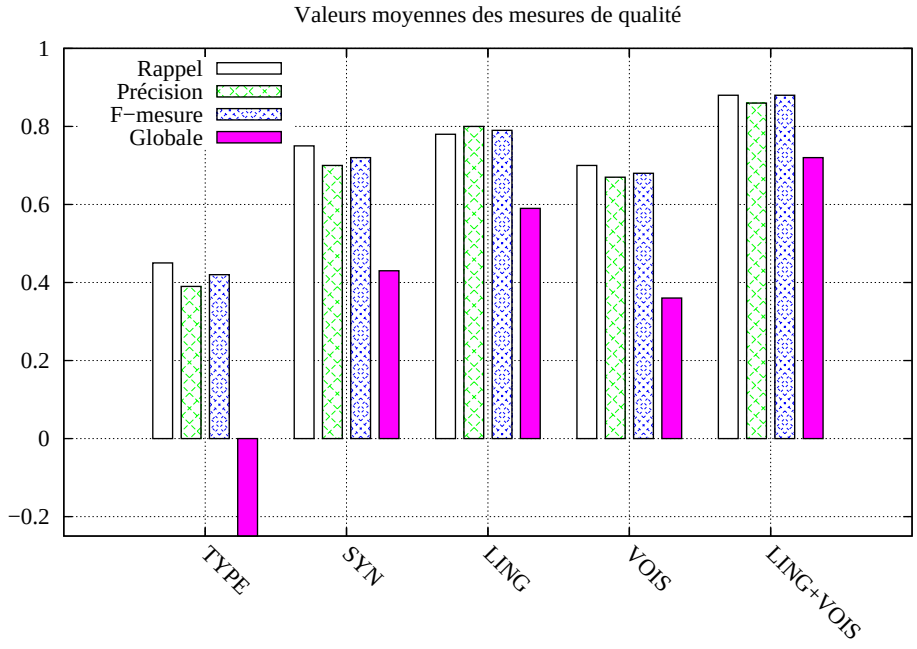


Figure 5.4 – Valeurs moyennes des mesures de qualité pour les différentes combinaisons d'appariement : Type (TYPE), Synonyme (SYN), Linguistique(LING=TYP+SYN), Voisinage (VOIS)

.

Les évaluations des systèmes de découverte de correspondances ont été faites de telle sorte qu'il est impossible de comparer directement leurs résultats. En effet, pour montrer l'efficacité de ces systèmes,

les évaluations ont été opérées en utilisant diverses méthodologies, ensembles de données et métriques, rendant ainsi difficile d'affirmer l'efficacité de chaque système.

Cependant, nous estimons que nos résultats sont prometteurs étant donné qu'il est difficile d'obtenir de telles valeurs en présence de schémas hétérogènes. En outre, ces résultats montrent que notre approche se comporte bien en comparaison aux approches précédentes (figure 5.5).

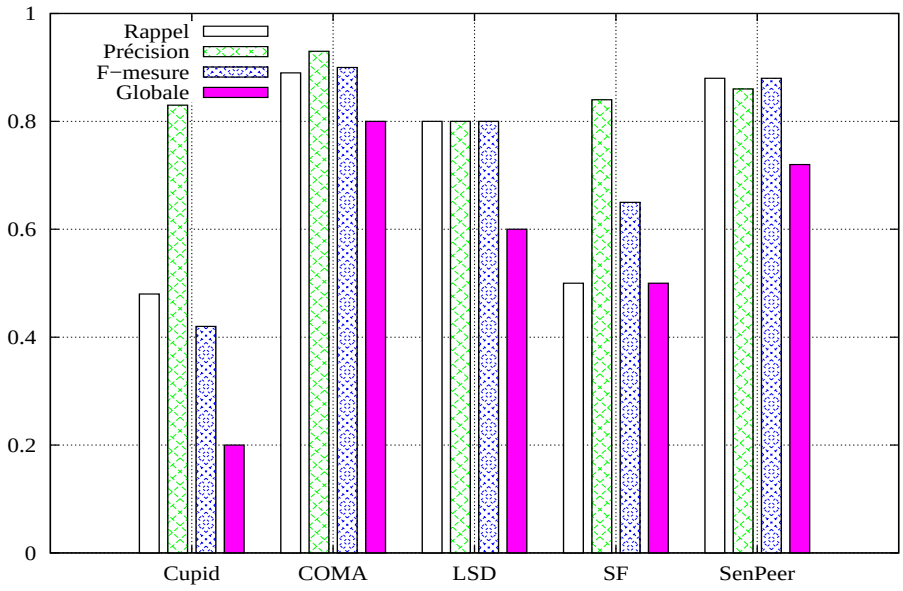


Figure 5.5 – Comparaison avec les résultats des autres systèmes

5.3 Evaluation du routage sémantique

Dans ce paragraphe, nous présentons l'évaluation du mécanisme de routage sémantique. Toutes les expériences ont été conduites sur un Pentium IV avec une fréquence 2.90 GHz, une mémoire de 512 Mo et un disque dur de 40 GB tournant sous le système Windows XP.

5.3.1 Outils de simulation

Les systèmes P2P ne sont pas initialisés et maintenus par une autorité centrale. Par conséquent, créer un réseau et mesurer ses performances est une tâche difficile. La simulation d'événements discrets peut aider à comprendre le comportement du système. Plusieurs projets de recherche tels que Freenet[26] et Anthill[79] ont utilisé la simulation dans le but de montrer leurs performances. La simulation d'événements discrets permet d'observer le comportement du système. Le modèle a un état décrit par des variables qui définissent de façon complète les caractéristiques du système[109]. L'état du modèle est souvent encapsulé dans un ensemble d'entités (objets en programmation orienté objets). Les événements discrets changeant l'état du système se produisent à différents points dans le temps (par opposition au changement continu d'états). Les événements peuvent déclencher de nouveaux événements. Des variables statistiques définissent alors les mesures de performance utiles à l'utilisateur.

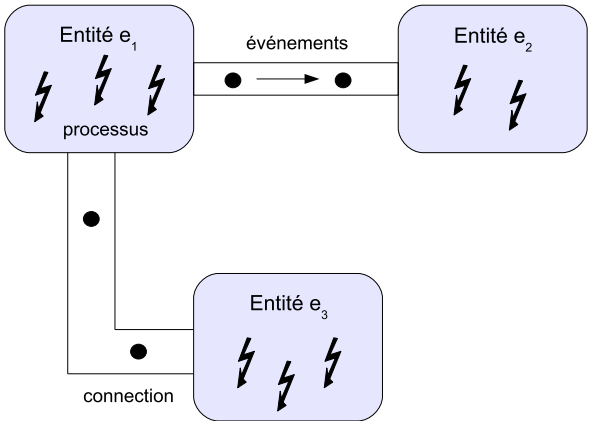


Figure 5.6 – Abstraction dans les paquetages de Simulation d'événements discrets[109].

Comme simulateur d'événements discrets nous utilisons SimJava[61]. SimJAVA est ensemble d'outils permettant de construire des modèles de systèmes complexes. Le paquetage SimJAVA a été conçu pour simuler des réseaux assez statiques d'entités actifs qui communiquent en s'envoyant des objets via des ports. Il inclue les services suivants :

- abstraction des entités ;
- connections entre les entités ;
- les événements transmis sur ces ports ;
- abstraction des processus tournant sur les entités.

5.3.2 Paramètres d'évaluation

5.3.2.1 Configuration de la topologie initiale du réseau

La simulation a été initialisée en répandant plusieurs instances de pairs et de super-pairs sur la même machine. Nous considérons un système avec 10 super-pairs chacun avec une communauté et un nombre de pairs allant de 100 à 3000. Chaque pair se connecte au hasard à un super-pair qui est son point d'accès au réseau et qui va l'aider à trouver le super-pair responsable de sa communauté. Dans la suite nous ne faisons plus de supposition sur la topologie du réseau.

5.3.2.2 Génération des expertises et des requêtes dans les expériences.

Les expertises des super-pairs sont générées en accord avec leurs communautés sémantiques. Quand un pair joint le réseau, la partie de son expertise qui est subsumée par celle de l'expertise de son super-pair parrain est détectée et elle est stockée de telle sorte qu'il soit possible de la réutiliser pour le routage des requêtes. Les requêtes sont formulées au hasard sur les schémas des pairs à la première exécution et réutilisées pour toutes les autres exécutions avec des réseaux de tailles plus importantes. Les pairs évaluent d'abord les requêtes sur leurs schémas locaux et les envoient ensuite à leurs super-pairs qui vont appliquer l'algorithme de sélection de (super-)pairs pertinents en faisant la mise en correspondance de chaque message requête avec les expertises détenues.

5.3.3 Mesures.

Pour évaluer notre mécanisme de routage, nous utilisons des métriques telles que le rappel et la précision, empruntées à la Recherche d'Informations, le nombre de messages par trace de requête et le temps de réponse. Nous les appliquons au niveau de la sélection des pairs pertinents. Étant donné une requête Q, soit $S_{trouves}$ l'ensemble des pairs sélectionnés et $S_{pertinents}$ l'ensemble des pairs pertinents, les mesures sont les suivantes.

Rappel. Pour une requête Q, le Rappel R_{Pair} mesure la proportion de pairs ayant des données pertinentes pour Q qui ont été sélectionnés.

$$R_{Pair} = \frac{|S_{pertinents} \cap S_{trouves}|}{|S_{pertinents}|} \tag{5.9}$$

Précision. Pour une requête Q, la Précision P_{Pair} indique combien parmi les pairs sélectionnés ont des données pertinentes.

$$P_{Pair} = \frac{|S_{pertinents} \cap S_{trouves}|}{|S_{trouves}|}. \tag{5.10}$$

Nombre de messages. Ce paramètre est lié à la charge du réseau. Il indique combien de messages sont générés pour répondre à une requête.

Temps de réponse. Le temps de réponse indique la rapidité avec laquelle on obtient une réponse.

5.3.4 Résultats expérimentaux

Dans nos expériences, nous avons simulé avec les deux configurations suivantes:

- ***Configuration A* (CONF A):** Les pairs sont regroupés au hasard autour de super-pairs. Le super-pair sélectionne au hasard un ensemble de pairs à qui envoyer la requête.
- ***Configuration B* (CONF B) :** Les pairs sont organisés en communautés sémantiques autour de super-pairs en accord avec leurs thèmes d'intérêts. Nous appliquons notre sélection basée sur l'expertise.

Nous avons mesuré le temps de réponse moyen (Figure 5.7), le nombre moyen de messages par trace de requête (Figure 5.8) ainsi que le rappel et la précision pour les configurations *CONF A* et *CONF B* pour des réseaux de différentes tailles. Les points dans tous les graphes représentent la moyenne pour un certain nombre de pairs (requêtes). Le nombre de pairs contactés pour répondre à une requête affecte le temps de réponse ainsi que le nombre de messages. Le passage à l'échelle du système peut être réalisé seulement si le nombre de tels pairs peut être réduit. La topologie sémantique, ainsi que la sélection basée sur l'expertise permettent d'élaguer l'espace de recherche en n'envoyant les requêtes qu'aux pairs dont l'expertise est similaire au sujet de la requête, limitant ainsi l'inondation du réseau par les messages. Le temps de réponse a été mesuré indépendamment dans différentes exécutions. Ce temps est le temps de communication pris de la formulation de la requête à la réception des réponses en incluant le temps pris par les super-pairs pour traiter la requête, contacter les (super-)pairs pertinents et fusionner les réponses venant de ces derniers.

Dans la figure 5.8 L'écart entre les deux temps de réponse et le nombre de messages pour les deux configurations est très faible pour un réseau de taille inférieure à 500 pairs. Ceci est dû au fait que dans la configuration *CONF A* les pairs sont aussi regroupés et l'inondation est faite seulement entre super-pairs. Si la taille du réseau augmente d'un facteur de 10, le temps de réponse progresse d'un facteur de 3% et

5% pour *CONF B* et *CONF A* respectivement. Cet écart devient très significatif parce que si la taille du réseau augmente d'un facteur de 60, le temps de réponse augmente d'un facteur de 40 pour *CONF A* et 16 pour *CONF B*. Nous observons le même phénomène pour le nombre de messages (figure 5.8).

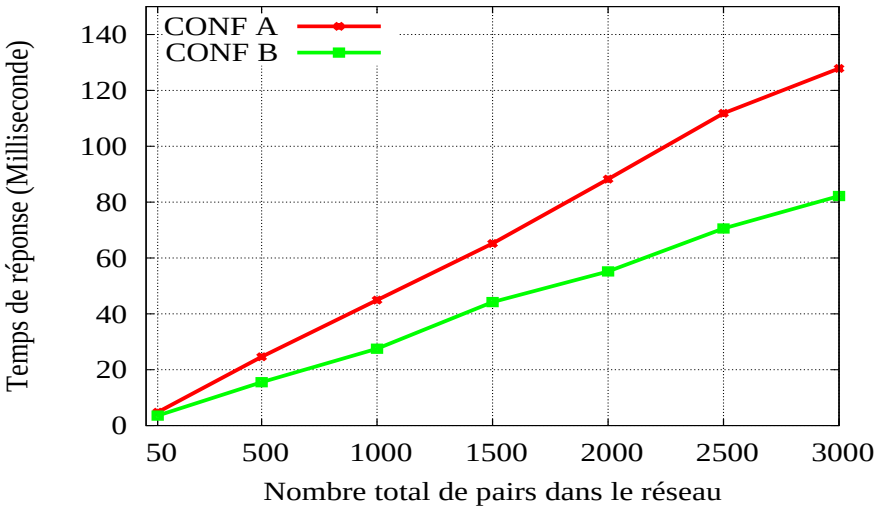


Figure 5.7 – Temps de réponse.

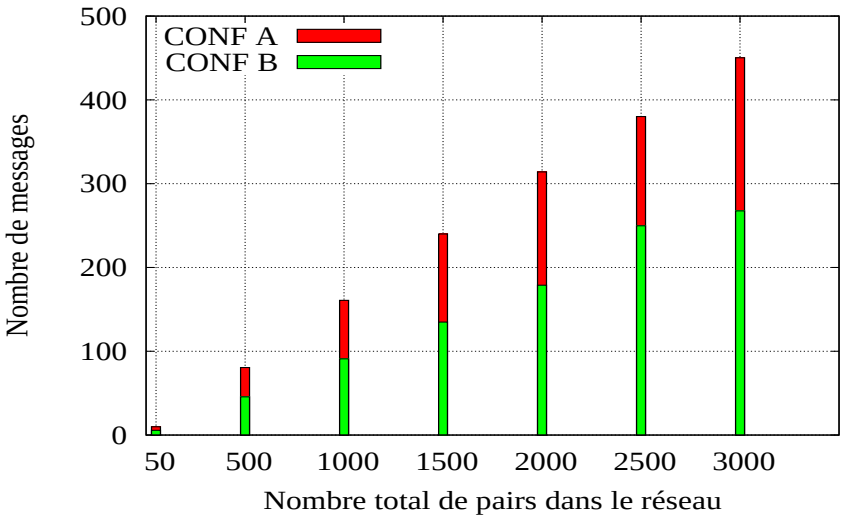


Figure 5.8 – Nombre de Messages.

La Figure 5.9 indique que dans *CONF A*, le rappel montre un comportement constant indépendamment de la taille du réseau, approximativement 25%. Le rappel de *CONF B* augmente avec la taille du réseau et atteint un pourcentage de presque 60% pour un réseau de 3000 pairs. En considérant le nombre de messages par trace de requête, nous remarquons que dans *CONF B*, pour atteindre un rappel de 60%, il faut moins de 200 messages comparé à *CONF A* qui nécessite plus de 450 messages pour atteindre un rappel de seulement 25%. Figure 5.10 montre que la sélection des pairs pertinents dans *CONF A* résulte en une faible précision de 1.5% qui diminue quand le nombre de pairs augmente, en dépit du

nombre élevé de messages. A l'opposé, dans *CONF B*, la précision est améliorée par la sélectivité de l'algorithme de sélection des pairs. Parallèlement, le nombre de messages diminue car les requêtes sont envoyées seulement au pairs prometteurs.

Au regard de ces résultats, on peut inférer que la topologie sémantique induite par les correspondances sémantiques intra et inter-communauté combinée avec la sélection basée sur l'expertise fournit des résultats prometteurs en termes de rappel, de précision. En définitive elle réduit aussi les efforts de traitement des requêtes et peut contribuer à l'amélioration des performances d'un PDMS.

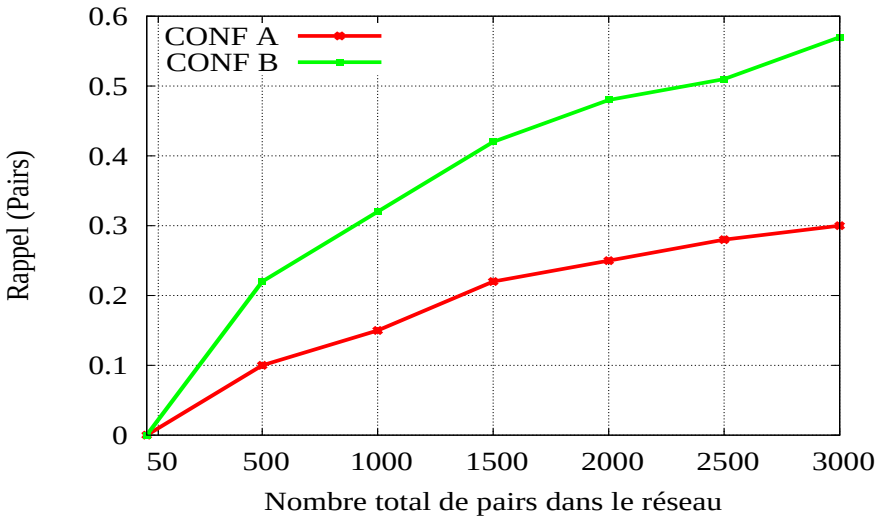


Figure 5.9 – Rappel.

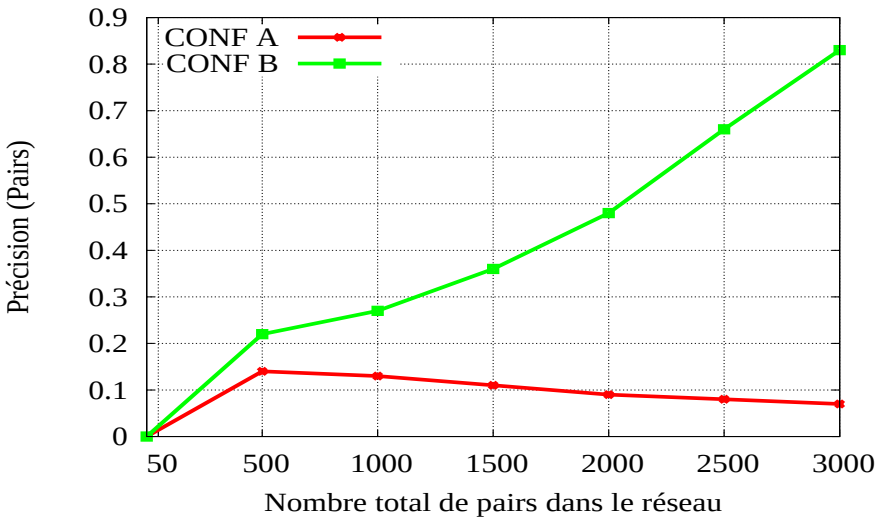


Figure 5.10 – Précision.

5.4 Optimisation de requêtes

Étant donné que les problèmes émanant du traitement de requêtes distribuées sont délaissés aux super-pairs, il est important de savoir si un super-pair est très occupé ou surchargé par l'exécution de certains opérateurs, ce qui pourrait causer un déséquilibre dans la distribution de la charge au sein du PDMS et ralentir l'exécution des requêtes. Pour ces raisons, chaque module d'optimisation doit être en mesure de collecter les informations sur la charge du super-pair local, mais aussi celles des (super-)pairs impliqués dans la requête, en évaluant leur charge de mémoire et de CPU avant de leur envoyer un sous-plan de requête.

Ainsi donc, des plans de requêtes alternatifs peuvent être générés si les (super-)pairs s'informent périodiquement de leurs charges de travail. Chaque (super-)pair mesure sa charge CPU et de mémoire et l'envoie aux super-pairs susceptibles de le solliciter pour le traitement d'une requête. Ces informations sont échangées de façon périodique entre les (super-)pairs pour prévenir d'une surcharge globale du PDMS. En utilisant cette information, le module d'optimisation de chaque super-pair peut décider si un sous-plan doit être envoyé à un voisin ou si un plan de requête alternatif doit être généré dans le but de produire des résultats en moins de temps. Cette information sur la charge est prise en compte dans notre modèle de coût introduit dans le chapitre précédent.

Cependant, il est difficile d'évaluer précisément la charge parce que sa valeur est valide au déploiement et non à l'exécution. En effet, la charge d'un super-pair peut avoir varié entre le moment où la requête est optimisée et celui où elle est exécutée. Nous approchons la charge en comptant le nombre de requêtes en cours d'exécution et celles qui sont en attente d'exécution.

Nous supposons que chaque (super-)pair connaît sa capacité maximale C_{max} et mesure la charge courante L_{SP} par le taux de requêtes entrant et sortant, mais aussi par celles en cours d'exécution à son niveau. La charge est donc constituée de deux paramètres :

- Les messages constitués par les requêtes qui sont en cours d'exécution. Ces requêtes représentent la charge courante mais elles n'auront plus d'effet si elles arrivent à terme avant l'exécution de la nouvelle requête.
- Les messages arrivant au (super-)pair. Les requêtes correspondantes ne contribuent pas encore à la charge de travail, mais le feront une fois en cours d'exécution.
- Les messages sortant qui ont pour effet d'alléger la charge.

Soit N_c le nombre de requêtes courantes, N_s le nombre de requêtes sortant du nœud et N_e le nombre de requêtes arrivant au (super-)pair. Pour estimer l'influence de chaque paramètre nous avons lancé des simulations avec les stratégies suivantes :

- SANS : La charge n'est pas prise en compte dans l'évaluation du coût ($L_{SP} = 0$).
- Cas 1 (CSE) : $L_{SP} = (N_c - N_s + N_e)/C_{max}$. La première stratégie prend en compte tous les types de messages.
- Cas 2 (CS) : $L_{SP} = (N_c - N_s)/C_{max}$. La seconde stratégie ignore l'impact des requêtes arrivant qui sont en attente d'exécution
- Cas 3 (CE) : $L_{SP} = (N_c + N_e)/C_{max}$. La troisième stratégie ne considère pas les messages sortant

Pour comparer l'impact de ces stratégies de prise en compte de la charge de travail dans l'estimation des coûts de plans nous avons lancé des expérimentations pour chaque cas en envoyant 15 requêtes à la fois au même super-pair qui doit partager sa capacité de travail entre les pairs le sollicitant. Pour chaque cas nous avons mesuré le temps d'exécution des requêtes.

La figure 5.11 montre les temps d'exécution enregistrés. Nous pouvons déjà noter que dans le cas SANS ne prenant pas en compte la charge, le temps de réponse est plus important et atteint son point culminant quand le nombre de requêtes présentes simultanément est maximal. Les configurations prenant

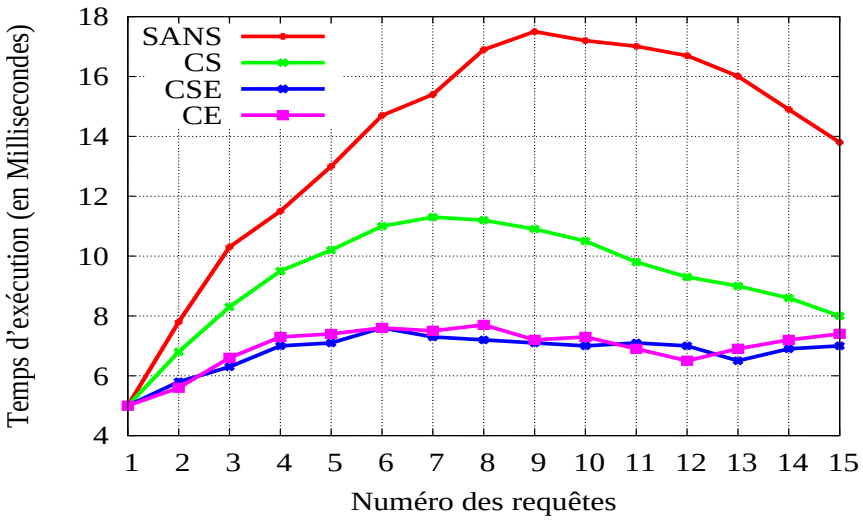


Figure 5.11 – Effet de la charge

en compte la charge offrent des résultats meilleurs. Le cas CSE se comporte beaucoup mieux que CS et est légèrement meilleur que CE, qui est lui même meilleur que le cas CS.

La prise en compte de la charge est donc importante dans l'optimisation des requêtes dans notre PDMS. Elle peut être prise en compte en divisant le coût de traitement par un facteur représentant l'effort supplémentaire à fournir et dépendant des requêtes en cours d'exécution, entrant dans le super-pair ou sortant de ce dernier. Ceci permet donc un raffinement simple du modèle de coût aboutissant à de meilleurs plans de requêtes et donc à une réduction des temps d'exécution des requêtes, contrairement à une situation sans prise en compte de la charge.

5.5 Conclusion

Dans ce chapitre, nous avons décrit les expérimentations en vue de l'évaluation des techniques proposées dans les chapitres précédents. Les expérimentations sur le processus de découverte de correspondances avec des données synthétiques ont révélé que les combinaisons d'appariement sont de meilleures qualités que les appariements uniques et que le processus de réconciliation est prometteur étant donné qu'il est difficile d'obtenir de telles valeurs en présence de schémas hétérogènes. La topologie sémantique induite par les correspondances sémantiques intra et inter-communauté, combinée avec la sélection basée sur l'expertise fournit des résultats prometteurs en termes de rappel, de précision mais aussi réduit le temps de réponse et le nombre de messages. En définitive, elle diminue aussi les efforts de traitement des requêtes et peut contribuer à l'amélioration des performances d'un PDMS. La prise en compte de la charge courante au sein des (super-)pairs durant la phase de génération des plans alternatifs permet de raffiner de façon simple le modèle de coût et d'obtenir des plans de requêtes à moindre coût. La façon de mesurer la charge a un impact sur le modèle de coût et donc sur la sélection des meilleurs plans.

CHAPITRE 6
Conclusion

Dans cette thèse, nous nous sommes intéressés au problème du partage de données (semi-)structurées et sémantiquement riches dans un contexte P2P. Dans un système P2P, un nombre important de nœuds mettent en commun leurs capacités de calcul et/ou leur données dans le but de se les partager. Les systèmes P2P peuvent être vus comme une extension des systèmes client/seveur classiques dans lesquels chaque nœud se comporte à la fois comme un client et un serveur. Ces systèmes présentent une caractéristique essentielle qui est l'autonomie car non seulement un pair quelconque peut joindre ou quitter le réseau à tout moment, mais le système devrait pouvoir continuer à fonctionner après chaque événement de ce type. De plus, ils présentent des propriétés essentielles telles que la décentralisation et la capacité à s'auto-organiser.

L'essor fulgurant du Web, des systèmes P2P, mais aussi les limites des bases de données distribuées, entre autres en terme de passage à l'échelle, ont fait de l'intégration des systèmes d'informations distribués à grande échelle l'un des domaines de recherche les plus importants en Informatique. Dans le même ordre d'idées, les systèmes P2P de gestion de données distribuées dénommés PDMS ont été introduits. Ils combinent les avantages des systèmes P2P avec ceux des bases de données distribuées en vue d'un partage à grande échelle de données sémantiquement riches. Bien vrai qu'un PDMS est basé sur l'architecture P2P, il est différent des systèmes classiques de partage de fichiers. En effet, la jonction d'un PDMS est plus lourde que celle d'un système de partage de fichiers P2P étant donné que des relations sémantiques doivent d'abord être spécifiées avant le processus d'interrogation du système.

Nous avons introduit le problème du partage de données dans les PDMS et nous avons aussi identifié les exigences pour un tel réseau de partage de données. Cependant, il apparaît qu'il n'existe pas une solution standard à ce problème d'intégration dans le cadre des systèmes pair-à-pair de gestion de données distribuées. En effet, il existe une panoplie de systèmes et d'approches, chacun avec ses compromis et ses spécificités et, par conséquent, elles sont difficilement comparables de façon stricte. Nous avons passé en revue l'état de l'art des travaux de recherche dans le domaine des PDMSs et les approches existantes ont été classifiées suivant les dimensions à prendre en compte dans la mise en place d'un PDMS. La contribution principale de cette thèse est la mise en place d'une infrastructure pour un PDMS permettant le partage de données sémantiquement riches en présence de plusieurs modèles de données et langages de requêtes et dans lequel les pairs partageant des thèmes d'intérêts similaires sont organisés en communautés sémantiques . En particulier les réalisations suivantes peuvent être identifiées :

- Nous avons proposé un processus de médiation sémantique dans un contexte P2P en présence de plusieurs modèles de données et en l'absence de schéma global. Un modèle pivot permettant d'exporter la connaissance sémantique des pairs mais aussi de faciliter la découverte des correspondances sémantiques entre schémas de pairs a été introduit. Les mesures de similarité sémantique permettant d'affirmer ces correspondances sémantiques ont aussi été définies en tenant compte des informations linguistiques sur les éléments des schémas mais aussi de leurs structures. La notion d'expertise de pair, en combinaison avec les matrices de correspondance sont à la base d'une topologie sémantique, ayant pour support des communautés sémantiques, au dessus de la topologie physique.

- Nous avons introduit un mécanisme de routage sémantique intra et inter-communauté s'appuyant sur la topologie sémantique et permettant d'envoyer les requêtes qu'aux sous-ensembles de pairs susceptibles d'y répondre correctement. Ceci permet de diminuer de façon significative les efforts de traitement de requêtes.
- Nous avons décrit un processus de traitement de requêtes dans lequel les requêtes sont échangées grâce à un format commun d'échange de requêtes et reformulées entre les schémas des pairs en tenant compte de la diversité des vocabulaires et des langages d'interrogation. Les plans de requêtes sont générés à partir des informations renvoyées par l'algorithme de routage sémantique et leur optimisation dépend d'un modèle de coût prenant en compte un certain nombre de paramètres liés à la nature du réseau du P2P. Cette optimisation est initiée au sein d'une communauté et elle est poursuivie dans les communautés liées et impliquées dans le traitement de la requête, rendant ainsi les phases de génération de plan et de routage imbriquées.
- Pour évaluer nos algorithmes, nous avons implémenté un simulateur pour notre PDMS. Nous avons conduit des expériences en rapport avec les techniques de découverte de correspondances, mais aussi sur l'impact de l'organisation des pairs en communautés sémantiques sur la distribution des requêtes intra et inter-communautés.

Limites et travaux futurs

Cette thèse a abordé deux aspects importants dans la conception d'un PDMS, notamment la réconciliation sémantique et le traitement des requêtes, Cependant, vu la complexité d'un PDMS, il nous a été impossible de couvrir à la fois tous les aspects liés à la conception d'un PDMS. Par exemple, nous n'avons pas pu démontrer les techniques proposées sur les données d'une application pratique. Nous projetons de le faire en implémentant les techniques proposées au dessus d'une application basée sur JXTA et manipulant les données du fleuve sénégal.

L'algorithme de découverte de correspondances proposé considère les correspondances *un-à-un*. Cependant, il peut exister des correspondances, *un-à-plusieurs* ou *plusieurs-à-plusieurs*. Supposons par exemple un schéma source contenant le nœud Nom et un schéma cible incluant les nœuds $Prenom$ et Nom. Nom ne devrait pas correspondre seulement à $Prenom$ mais aussi à Nom dans une correspondance *un-à-plusieurs*. En conséquence, l'information dans le nœud Nom est décomposée en $Prenom$ et Nom. Dans le cas opposé, $Prenom$ et Nom dans un schéma source pourraient correspondre à Nom dans une relation *plusieurs-à-un* et $Prenom$ et Nom devraient être fusionnés en Nom. Pour des transformations plus avancées telles que la fusion et la décomposition, il est nécessaire de considérer ces types de correspondances complexes. Comme travail futur, nous allons étudier un algorithme de découverte de correspondances dans notre contexte qui permet d'établir des correspondances complexes associées à des opérations de transformations telles que la fusion et la décomposition.

Nous projetons d'introduire un mécanisme pour déduire les correspondances transitives. Par exemple s'il y a une correspondance entre A et B et une autre entre B et C, il devrait y avoir aussi une correspondance entre A et C qui peut être dérivée des correspondances existantes.

Le choix adéquat des paramètres est très important pour un processus de découverte de correspondances sémantiques. Dans SenPeer, il y a cinq paramètres (quatre pour les poids des différents types de similarités, un seuil de similarité). Une méthode systématique d'apprentissage de ces paramètres au fil des processus de découverte peut améliorer de façon significative les résultats.

Une approche alternative pour la découverte des correspondances sémantiques dans un contexte P2P est de distribuer le calcul des similarités entre les pairs potentiellement riches en ressources. Nous uti-

lisons déjà ce procédé pour la découverte des correspondances entre les parrains de communautés sémantiquement liées. Il serait intéressant de l'appliquer au sein d'une même communauté en utilisant les capacités des pairs inactifs qui sont dans la même communauté et qui présentent des capacités de calcul pouvant supporter l'algorithme de découverte de correspondances. La distribution des tâches de calcul au sein de la même communauté peut aider à améliorer l'efficacité de la découverte de correspondances. Si les $sGraph$ sont partitionnés et distribués au différents pairs pour réaliser l'appariement, ceci peut effectivement réduire le temps d'exploration des $sGraph$.

Nous projetons d'étudier des techniques avancées permettant l'adaptabilité des plans de requêtes durant leur exécution. Un processus de traitement de requête présente la propriété d'adaptabilité si :

- il reçoit des informations de son environnement,
- il utilise ces informations pour déterminer son comportement et
- ce processus est répété au fil du temps, produisant ainsi un lien de dépendance entre environnement et comportement.[100]

Ces techniques de traitement de requêtes permettront de modifier un plan de requête au cours de son exécution. La raison principale de procéder de la sorte est que les informations sur l'environnement d'exécution peuvent ne plus correspondre aux informations qui ont été utilisées pendant l'optimisation. Le traitement de requêtes adaptatif consiste à (i) identifier le plan courant qui ne se comporte pas comme attendu et (ii) à basculer dans un autre plan qui est cette fois-ci meilleur.

Bibliographie

[1] Karl ABERER, Philippe CUDRÉ-MAUROUX et Manfred HAUSWIRTH. The chatty web: emergent semantics through gossiping. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 197–206, New York, NY, USA, 2003. ACM Press.

[2] Karl ABERER, Philippe CUDRÉ-MAUROUX, Manfred HAUSWIRTH et Tim Van PELT. Gridvine: Building internet-scale semantic overlay networks. In Sheila A. MCILRAITH, Dimitris PLEXOUSAKIS et Frank van HARMELEN, réds., *International Semantic Web Conference*, Lecture Notes in Computer Science, pages 107–121. Springer.

[3] Karl ABERER, Philippe CUDRÉ-MAUROUX, Anwitaman DATTA, Zoran DESPOTOVIC, Manfred HAUSWIRTH, Magdalena PUNCEVA et Roman SCHMIDT. P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, 32(3):29–33, 2003.

[4] Lada A. ADAMIC, Rajan M. LUKOSE, Amit R. PUNIYANI et Bernardo A. HUBERMAN. Search in power-law networks. *Physical Review E*, 64:046135.

[5] Philippe ADJIMAN, Philippe CHATALIC, Francois GOASDOUÉ, Marie-Christine ROUSSET et Laurent SIMON. Somewhere in the semantic web. In *PPSWR 2005: International Workshop on Principles and Practice of Semantic Web Reasoning*, pages 1–16, 2005.

[6] Reza AKBARINIA, Vidal MARTINS, Esther PACITTI et Patrick VALDURIEZ. Replication and query processing in the appa data management system. In *International Workshop on Distributed Data and Structures (WDAS)*, pages 19–33. Carleton Scientific, 2004.

[7] Marcelo ARENAS, Vasiliki KANTERE, Anastasios KEMENTSIETSIDIS, Iluju KIRINGA, Renée J. MILLER et John MYLOPOULOS. The hyperion project: from data integration to data coordination. *SIGMOD Rec.*, 32(3):53–58, 2003.

[8] M. M. ASTRAHAN, M. W. BLASGEN, D. D. CHAMBERLIN, K. P. ESWARAN, J. N. GRAY, P. P. GRIFFITHS, W. F. KING, R. A. LORIE, P. R. MCJONES, J. W. MEHL, G. R. PUTZOLU, I. L. TRAIGER, B. W. WADE et V. WATSON. System r: relational approach to database management. *ACM Trans. Database Syst.*, 1(2):97–137, 1976.

[9] Wolf-Tilo BALKE, Wolfgang NEJDL, Wolf SIBERSKI et Uwe THADEN. Progressive distributed top-k retrieval in peer-to-peer networks. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 174–185, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Equipe BDISIC. Projet sic-web sénégal. In *Compte rendu du Workshop des 10 et 11 juin, Université Gaston Berger de Saint-Louis du Sénégal*, 2004.

[11] Zohra BELLAHSENE, Charalambos LAZINITIS, Peter J. MCBRIEN et Nikolaos RIZOPOULOS. ixpeer: Implementing layers of abstraction in p2p schema mapping using automed. In *(IWI'2) : 2nd Workshop on Innovations in Web Infrastructure, Edinburgh, UK*, May 2006.

[12] Zohra BELLAHSENE et Mark ROANTREE. Querying distributed data in a super-peer based architecture. In *DEXA'04: 15th International Conference on Database and Expert Systems Applications, Zaragoza (Spain)*, pages 296–305, 2004.

[13] Sonia BERGAMASCHI, Silvana CASTANO et Maurizio VINCINI. Semantic integration of semi-structured and structured data sources. *SIGMOD Rec.*, 28(1):54–59, 1999.

[14] Sonia BERGAMASCHI, Francesco GUERRA et Maurizio VINCINI. A peer-to-peer information system for the semantic web. In *AP2PC03 : Proceedings of the International Workshop on Agents and Peer-to-Peer Computing (AP2PC03) Melbourne, Australia, July 14*, pages 113–122, 2003.

[15] P. BERNSTEIN, F. GIUNCHIGLIA, A. KEMENTSIETSIDIS, J. MYLOPOULOS, L. SERAFINI et I. ZAIHRAYEU. Data management for peer-to-peer computing: A vision. In *Prooceedings of the Workshop on the Web and Databases, WebDB 2002.*, pages 89–94, 2002.

[16] Philip A. BERNSTEIN, Sergey MELNIK, Michalis PETROPOULOS et Christoph QUIX. Industrial-strength schema matching. *SIGMOD Rec.*, 33(4):38–43, 2004.

[17] Peter A. BONCZ et Caspar TREIJTEL. Ambientdb: Relational query processing in a p2p network. In *DBISP2P : Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P). Springer Verlag*, volume 2944, pages 153–168, 2003.

[18] Alexander BORGIDA et Luciano SERAFINI. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.

[19] Aida BOUKOTTAYA et Christine VANOIRBEEK. Schema matching for transforming structured documents. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, pages 101–110, New York, NY, USA, 2005. ACM Press.

[20] Michael BOYD, Sasivimol KITTIVORAVITKUL, Charalambos LAZANITIS, Peter MCBRIEN et Nikos RIZOPOULOS. Automed: A bav data integration system for heterogeneous data sources. In *CAISE'04 : The 16th International Conference on Advanced Information Systems Engineering, Riga, Latvia*, pages 82–97, 2004.

[21] Sapkota BRAHMANANDA. Design of peer-to-peer protocol for ambientdb. *Master's thesis, University of Twente*, 2003.

[22] Reinhard BRAUMANDL, Markus KEIDL, Alfons KEMPER, Donald KOSSMANN, Stefan SELTZSAM et Konrad STOCKER. Objectglobe:open distributed query processing services on the internet. *IEEE Data Engineering Bulletin*, 24(1):64–70, 2001.

[23] Jeen BROEKSTRA, Marc EHRIG, Peter HAASE, Frank van HARMELEN, Maarten MENKEN, Peter MIKA, Bjorn SCHNIZLER et Ronny SIEBES. Bibster : a semantics-based bibliographic peer-to-peer system. In *Proceedings of the 3rd International. Semantic Web Conference, Hiroshima, Japan*, pages 122–136, 2004.

[24] Min CAI, Martin FRANK, Jinbo CHEN et Pedro SZEKELY. Maan: A multi-attribute addressable network for grid information services. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 184, Washington, DC, USA, 2003. IEEE Computer Society.

[25] Silvana CASTANO, Alfio FERRARA et Stefano MONTANELLI. H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *Proceedings of the 1st VLDB International Workshop on Semantic Web and Databases (SWDB 2003) Berlin, Germany, September*, pages 231–250, 2003.

[26] Ian CLARKE, Oskar SANDBERG, Brandon WILEY et Theodore W. HONG. Freenet: a distributed anonymous information storage and retrieval system. In *International workshop on Designing privacy enhancing technologies*, pages 46–66, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[27] Arturo CRESPO et Hector GARCIA-MOLINA. Routing indices for peer-to-peer systems. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.

[28] Isabel F. CRUZ, Huiyong XIAO et Feihong HSU. Peer-to-peer semantic integration of xml and rdf data sources. In *Third International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*, pages 108–119, 2004.

[29] Neil DASWANI, Hector GARCIA-MOLINA et Beverly YANG. Open problems in data-sharing peer-to-peer systems. In *ICDT '03: Proceedings of the 9th International Conference on Database Theory*, pages 1–15, London, UK, 2003. Springer-Verlag.

[30] Hong Hai DO, Sergey MELNIK et Erhard RAHM. Comparison of schema matching evaluations. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, pages 221–237, London, UK, 2003. Springer-Verlag.

[31] Hong-Hai DO et Erhard RAHM. Coma - a system for flexible combination of schema matching approaches. In *Proc. 28th Conference on Very Large Databases (VLDB), Hongkong, August*, pages 610–621, 2002.

[32] AnHai DOAN, Pedro DOMINGOS et Alon Y. HALEVY. Reconciling schemas of disparate data sources: a machine-learning approach. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 509–520, New York, NY, USA, 2001. ACM Press.

[33] XML Schema Part 2: Datatypes Second EDITION. http://www.w3.org/tr/xmlschema-2/.

[34] David FAYE, Gilles NACHOUKI et Valduriez PATRICK. Senpeer : un système pair-à-pair de médiation de données. *Revue ARIMA Volume 4*, 2006.

[35] David FAYE, Gilles NACHOUKI et Patrick VALDURIEZ. Intégration de données hétérogènes dans senpeer. In *Conférence Africaine sur la Recherche en Informatique (CARI'06) Cotonou, Bénin*, pages 181–188, 2006.

[36] David FAYE, Gilles NACHOUKI et Patrick VALDURIEZ. Semantic query routing in senpeer, a p2p data management system. In *1st International Conference on Network-Based Information Systems Sept. 3 - 7, Regensburg, Germany*, 2007.

[37] Mary FERNANDEZ, Daniela FLORESCU, Jaewoo KANG, Alon Y. LEVY et Dan SUCIU. Strudel: A website management system. In *Proceedings ACM SIGMOD International Conference on Management of Data,Tucson, Arizona, USA,*, pages 549–552, 1997.

[38] Enrico FRANCONI, Gabriel M. KUPER, Andrei LOPATENKO et Ilya ZAIHRAYEU. The codb robust peer-topeer database system. In *Proceedings of the Twelfth Italian Symposium on Advanced Database Systems (SEBD 2004), S. Margherita di Pula, Cagliari, Italy*, pages 382–393, 2004.

[39] Marc FRIEDMAN, Alon LEVY et Todd MILLSTEIN. Navigational plans for data integration. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 67–73, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[40] Hector GARCIA-MOLINA, Jennifer WIDOM et Jeffrey D. ULLMAN. *Database System Implementation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[41] Alon HALEVY, Zacharia IVES, Jayant MADHAVAN, Peter MORK, Dan SUCIU et Igor TATARINOV. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):787–798, 2004.

[42] Alon Y. HALEVY. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[43] Alon Y. HALEVY. Data integration: A status report. In *BTW*, pages 24–29, 2003.

[44] Dennis HEIMBIGNER et Dennis MCLEOD. A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278, 1985.

[45] Sven HERSCHEL et Ralf HEESE. Humboldt discoverer: A semantic p2p index for pdms. In *Proceedings of the International Workshop Data Integration and the Semantic Web, Porto, Portugal, June*, 2005.

[46] Ryan HUEBSCH, Brent CHUN, Ryan M. HELLERSTEIN, Boon Thau LOO, Petros MANIATIS, Timothy ROSCOE, Scott SHENKER, Ion STOICA et Aydan R. YUMEREFENDI. The architecture of pier: an internet-scale query processor. In *Proceedings of the Second Biennial Conference on Innovative Data Systems Research,Asilomar, CA, USA*, pages 28–43, 2005.

[47] Ryan HUEBSCH, Ryan M. HELLERSTEIN, Nick LANHAM, Boon Thau LOO, Scott SHENKER et Ion STOICA. Querying the internet with pier. In *Proceedings of 29th International Conference on Very Large Data Bases, Berlin, Germany*, pages 321–332, 2003.

[48] Ryan HUEBSCH et Shawn R. JEFFREY. Freddies: Dht-based adaptive query processing via federated eddies. Rapport technique, EECS Department, University of California, Berkeley, Jul 2004.

[49] Ali R. HURSON, Simin H. PAKZAD et M. W. BRIGHT. *Multidatabase Systems: An Advance Solution for Global Information Sharing*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1993.

[50] Yannis E. IOANNIDIS et Younkyung KANG. Randomized algorithms for optimizing large join queries. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 312–321, New York, NY, USA, 1990. ACM Press.

[51] Yannis E. IOANNIDIS et Eugene WONG. Query optimization by simulated annealing. In *SIGMOD '87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pages 9–22, New York, NY, USA, 1987. ACM Press.

[52] Sam JOSEPH. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 202–214, London, UK, 2002. Springer-Verlag.

[53] Vanja JOSIFOVSKI et Tore RISCH. Integrating heterogenous overlapping databases through object-oriented transformations. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 435–446, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[54] Vanja JOSIFOVSKI, Peter SCHWARZ, Laura HAAS et Eileen LIN. Garlic: a new flavor of federated query processing for db2. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 524–532, New York, NY, USA, 2002. ACM Press.

[55] JXTA. www.jxta.org.

[56] Frans M. KAASHOEK et David R. KARGER. Koorde: A simple degree-optimal distributed hash table. In *Second International Workshop on Peer-to-Peer Systems, Berkeley, CA, USA*, pages 98–107, 2003.

[57] KAAZA. www.kazaa.com.

[58] Alfons KEMPER et Christian WIESNER. Hyperqueries: Dynamic distributed query processing on the internet. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 551–560, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[59] Giorgos KOKKINIDIS et Vassilis CHRISTOPHIDES. Semantic query routing and processing in p2p database systems: The ics-forth sqpeer middleware. In *EDBT Workshops, Heraklion, Crete, Greece*, pages 486–495, 2004.

[60] Donald KOSSMANN. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[61] Wolfgang KREUTZER, Jane HOPKINS et Marcel van MIERLO. Simjava - a framework for modeling queueing networks in java. In *Winter Simulation Conference*, pages 483–488.

[62] John KUBIATOWICZ, David BINDEL, Yan CHEN, Steven E. CZERWINSKI, Patrick R. EATON, Dennis GEELS, Ramakrishna GUMMADI, Sean C. RHEA, Hakim WEATHERSPOON, Westley WEIMER, Chris WELLS et Ben Y. ZHAO. Oceanstore : An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Cambridge, MA, USA.*, pages 190–201, 2000.

[63] Maurizio LENZERINI. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM Press.

[64] Vladimir I. LEVENSHTEIN. Binary codes capable of correcting deletions, insertions, and reversals. Rapport technique 8, 1966.

[65] Alon Y. LEVY, Anand RAJARAMAN et Joann J. ORDILLE. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996. VLDB Endowment, Saratoga, Calif.

[66] Wen-Syan LI et Chris CLIFTON. Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowledge Engineering.*, 33(1):49–84, 2000.

[67] Boon Thau LOO, Ryan HUEBSCH, Ion STOICA et Joseph M. HELLERSTEIN. The case for a hybrid p2p search infrastructure. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04)*, pages 141–150, San Diego, CA, February 2004.

[68] Alexander LÖSER, Christoph TEMPICH, Bastian QUILITZ, Steffen STAAB, Wolf Tilo BALKE et Wolfgang NEJDL. Searching dynamic communities with personal indexes. In *ISWC 2005 : Proceedings of the 4th International Semantic Web Conference, ISWC 2005*, Galway, Ireland.

[69] Alexander LÖSER, Martin WOLPERS, Wolf SIBERSKI et Wolfgang NEJDL. Efficient data store discovery in a scientific P2P network.

[70] Qin LV, Pei CAO, Edith COHEN, Kai LI et Scott SHENKER. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM Press.

[71] Jayant MADHAVAN, Philip A. BERNSTEIN et Erhard RAHM. Generic schema matching with cupid. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[72] Jayant MADHAVAN et Alon Y. HALEVY. Composing mappings among data sources. In *VLDB : Proceedinds of the 29th VLDB Conference, Berlin, Germany*, pages 572–583, 2003.

[73] Alexander MAEDCHE et Steffen STAAB. Measuring similarity between ontologies. In *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 251–263, London, UK, 2002. Springer-Verlag.

[74] Dahlia MALKHI, Moni NAOR et David RATAJCZAK. Viceroy: a scalable and dynamic emulation of the butterfly. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 183–192, New York, NY, USA, 2002. ACM Press.

[75] Shashidhar MERUGU, Sridhar SRINIVASAN et Ellen W. ZEGURA. Adding structure to unstructured peer-to-peer networks: the use of small-world graphs. *Journal of Parallel and Distributed Computing*, 65(2):142–153, 2005.

[76] René J. MILLER, Laura M. HAAS et Mauricio A. HERNANDEZ. Schema mapping as query discovery. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 77–88, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[77] Todd D. MILLSTEIN, Alon Y. LEVY et Marc FRIEDMAN. Query containment for data integration systems. In *PODS*, pages 67–75, 2000.

[78] Tova MILO et Sagit ZOHAR. Using schema matching to simplify heterogeneous data translation. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 122–133, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[79] Alberto MONTRESOR, Hein MELING et Özalp BABAOGLU. Towards adaptive, resilient and self-organizing peer-to-peer systems. In *Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 300–305, London, UK, 2002. Springer-Verlag.

[80] Moni NAOR et Udi WIEDER. A simple fault tolerant distributed hash table. In *Proceedings of the International Workshop on Peer-to-Peer Systems, Berkeley, CA, USA*, pages 88–97, 2003.

[81] NAPSTER. http://www.napster.com/.

[82] Wolfgang NEJDL, Boris WOLF, Changtao QU, Stefan DECKER, Michael SINTEK, Ambjörn NAEVE, Mikael NILSSON, Matthias PALMÉR et Tore RISCH. Edutella: a p2p networking infrastructure based on rdf. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 604–615, New York, NY, USA, 2002. ACM Press.

[83] Wolfgang NEJDL, Martin WOLPERS, Wolf SIBERSKI, Christoph SCHMITZ, Mario SCHLOSSER, Ingo BRUNKHORST et Alexander LÖSER. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 536–543, New York, NY, USA, 2003. ACM Press.

[84] Wee Siong NG, Beng Chin OOI, Kian-Lee TAN et Aoying ZHOU. Peerdb: A p2p-based system for distributed data sharing. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, pages 633–644, 2003.

[85] Mourad OUAZZANI. *Efficient Delivery of Web Services*. Thèse de Doctorat, Virginia Polytechnic Institute and State University, 2004.

[86] Tamer ÖZSU et Patrick VALDURIEZ. *Principles of distributed database systems (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[87] Luigi PALOPOLI, Giorgio TERRACINA et Domenico URSINO. Experiences using dike, a system for supporting cooperative information system and data warehouse design. *Information Systems*, 28(7):835–865, 2003.

[88] Vassilis PAPADIMOS, David MAIER et Kristin TUFTE. Distributed query processing and catalogs for peer-to-peer systems. In *Proceedings of the CIDR Conference*, 2003.

[89] Yannis PAPAKONSTANTINOU, Hector GARCIA-MOLINA et Jennifer WIDOM. Object exchange across heterogeneous information sources. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.

[90] Greg C. PLAXTON, Rajmohan RAJARAMAN et Andréa W. RICHA. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA '97: Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320, New York, NY, USA, 1997. ACM Press.

[91] Gnutella 0.6 protocol DRAFT.. http://rfc-gnutella.sourceforge.net/developer/testing/index.html.

[92] Erhard RAHM et Philip A. BERNSTEIN. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[93] Sylvia RATNASAMY, Paul FRANCIS, Mark HANDLEY, Richard KARP et Scott SCHENKER. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[94] Sean RHEA, Brighten GODFREY, Brad KARP, John KUBIATOWICZ, Sylvia RATNASAMY, Scott SHENKER, Ion STOICA et Harlan YU. Opendht: a public dht service and its uses. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 73–84, New York, NY, USA, 2005. ACM Press.

[95] John RISSON et Tim MOORS. Survey of research towards robust peer-to-peer networks: search methods. *Comput. Networks*, 50(17):3485–3521, 2006.

[96] Jordan RITTER. Why gnutella can't scale. Rapport technique, Darkridge, Inc., 2001.

[97] Andrea M. RODRIGUEZ et Max J. EGENHOFER. Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003.

[98] Antony I. T. ROWSTRON et Peter DRUSCHEL. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[99] Ozgur D. SAHIN, Abhishek GUPTA, Divyakant AGRAWAL et Amr El ABBADI. Query processing over peer-to-peer data sharing systems. Rapport technique, UCSB/CSD-2002-28, University of California at Santa Barbara, 2002.

[100] Arnaud SAHUGUET. *ubQL: A Distributed Query Language to Program Distributed Query Systems*. Thèse de Doctorat, University of Pennsylvania, 2002.

[101] Stefan SAROIU, Krishna P. GUMMADI et Steven D. GRIBBLE. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, January 2002.

[102] Nima SARSHAR et Vwani ROYCHOWDHURY. Scale-free and stable structures in complex ad hoc networks. *Physical Review E*, 69:026101.

[103] Carlo SARTIANI, Paolo MANGHI, Giorgio GHELLI et Giovanni CONFORTI. Xpeer: A self-organizing xml p2p database system. In *Proceedings of the EDBT 2004 Workshop on Peer-to-Peer Computing and Databases, Heraklion, Crete, Greece*, pages 456–465, 2004.

[104] Christoph SCHMITZ. Self-organizing a small world by topic. In *Proceedings of the MobiQuitous'04 Workshop on Peer-to-Peer Knowledge Management (P2PKM 2004), Boston,MA, USA*, pages 303–310, 2004.

[105] Michael STEINBRUNN, Guido MOERKOTTE et Alfons KEMPER. Heuristic and randomized optimization for the join ordering problem. *The VLDB Journal*, 6(3):191–208, 1997.

[106] Ion STOICA, Robert MORRIS, David KARGER, Frans KAASHOEK et Hari BALAKRISHNAN. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[107] Michael STONEBRAKER, Paul M. AOKI et Robert DEVINE. Mariposa: A new architecture for distributed data. Rapport technique, Berkeley, CA, USA, 1993.

[108] Christoph TEMPICH, Steffen STAAB et Adrian WRANIK. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 640–649, New York, NY, USA, 2004. ACM Press.

[109] Christoph TEMPICH, Julien TANE, Steffen STAAB, Marc EHRIG et Christoph SCHMITZ. Towards evaluation of peer-to-peer-based distributed information management systems. In L. van Elst et al. (EDS.), réd., *Agent-mediated Knowledge Management - AMKM-2003, AAAI Spring Symposium 2003, Stanford, March 24-26*, pages 73–88, 2003.

[110] Anthony TOMASIC, Louiqa RASCHID et Patrick VALDURIEZ. Scaling heterogeneous databases and the design of disco. In *International Conference on Distributed Computing Systems*, pages 449–457, 1996.

[111] Amos TVERSKY. Features of similarity. In *Psychological Review*, volume 84, pages 327–352, 1977.

[112] Patrick VALDURIEZ et Esther. PACITTI. Data management in large-scale p2p systems. In *Int. Conf. on High Performance Computing for Computational Science (VecPar'2004)*, pages 109–122. LNCS 3402, Springer, 2004.

[113] Son VUONG et Juan LI. Efa: An efficient content routing algorithm in large peer-to-peer overlay networks. In *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 216–224, Washington, DC, USA, 2003. IEEE Computer Society.

[114] I Wayan Simri WICAKSANA. *A Peer-to-peer (P2P) Based Semantic Agreement Approach for Spatial Information Interoperability*. Thèse de Doctorat, Gunadarma University, 2006.

[115] Li XU et David W. EMBLEY. Combining the best of global-as-view and local-as-view for data integration. In *ISTA*, pages 123–136, 2004.

[116] Beverly YANG et Hector GARCIA-MOLINA. Improving search in peer-to-peer networks. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 5–14, Washington, DC, USA, 2002. IEEE Computer Society.

[117] Mikalai YATSKEVICH, Fausto GIUNCHIGLIA et Paolo AVESANI. A large scale dataset for the evaluation of matching systems. Rapport technique, Department of Information and Communication Technology, University of Trento, 2006.

[118] Yongluan ZHOU, Beng Chin OOI, Kian-Lee TAN et Wee Hyong TOK. An adaptable distributed query processing architecture. *Data Knowledge. Engeenering*, 53(3):283–309, 2005.

Liste des tableaux

2.1 Comparaison des approches GAV, LAV et GLAV 13
2.2 Avantages et inconvénients des systèmes P2P (comparés au modèle Client/Serveur) . . . 16
2.3 Les PDMS vus sous différents aspects. 29
2.4 Classification des systèmes du point de vue routage de requêtes et génération de plans . . 30

3.1 Grammaire pour les étiquettes des nœuds. 46
3.2 Exemple de similarités de types de données. 55
3.3 Matrice de correspondance Super-pair/pair. 57

4.1 Définition de la classe représentant un nœud du format d'échange de requêtes. 72
4.2 Liste des opérateurs algébriques 84

5.1 Caractéristiques des schémas testés. 102
5.2 Matrice de confusion. 104
5.3 Valeurs des paramètres pour l'appariement. 105

Table des figures

1.1 Partage de données du fleuve en présence de plusieurs modèles de données. 4

2.1 Phases du traitement de requêtes dans un DDMS. 10
2.2 Un exemple GAV. 12
2.3 Un exemple LAV. 12
2.4 Un exemple GLAV. 13
2.5 Taxonomie des techniques de découverte de correspondances sémantiques entre schémas. 14
2.6 Différentes approches P2P 17
2.7 Indexation Locale. 18
2.8 Indexation centralisée. 18
2.9 Réseaux avec Raccourcis. 19
2.10 Inondation avec filtrage. 19
2.11 Indexation avec Super-pair. 20
2.12 Indexation dans les réseaux Pair-à-Pair. 21
2.13 Combinaison entre schémas et distribution[83]. 22

3.1 Partie de la taxonomie organisant les données de la vallée du fleuve sénégal. 38
3.2 Réseau Super-pair sémantique organisé par thèmes. 39
3.3 Architecture d'un pair. 40
3.4 Architecture d'un super-pair. 42
3.5 Procédure de création d'un canal de communication bidirectionnel 43
3.6 Une architecture de médiation à deux niveaux. 45
3.7 $sGraph$ partiel sur les cultures pratiquées sur les sols du bassin du fleuve. 48
3.8 Définition de la classe représentant un nœud du modèle interne $sGraph$. 49
3.9 Processus de découverte de correspondances entre schémas. 51
3.10 Combinaison de plusieurs *apparieurs*. 52
3.11 Voisinage sémantique du nœud $culture\_irriguée$ 56
3.12 Médiation de données des domaines Agriculture et Pédologie. ISRA(Institut Sénégalais de Recherche Agronomique), SAED (Société d'Aménagement et d'Exploitation du Delta). 59
3.13 Adhésion à une communauté. 64
3.14 Fondation des communautés sémantiques durant le processus de découverte de correspondances. 65

4.1 Principales étapes du processus d'interrogation. 70
4.2 Une requête SQL et la requête SQUEL correspondante. 72
4.3 Sélection basée sur l'expertise. 75
4.4 Annotation sémantique de requête. 76
4.5 Requêtes réécrites. 80
4.6 Plan d'exécution pour la requête Q du pair SAED. 86
4.7 Architecture du module d'optimisation de requêtes d'un (Super-)pair. 87

4.8 Plan de requête équivalent au plan QEP de la figure 4.6. 90
4.9 Diagramme d'état du processus d'optimisation à deux phases. 91
4.10 Traitement de requêtes distribuées. Instanciation des plans locaux et distribution des plans distants 99

5.1 Extrait du $sGraph$ de l'*Hydrologie* 102
5.2 Extrait du $sGraph$ des *Activités Hydro-agricoles*. 103
5.3 Classification des correspondances sémantiques. 104
5.4 Valeurs moyennes des mesures de qualité pour les différentes combinaisons d'appariement : Type (TYPE), Synonyme (SYN), Linguistique(LING=TYP+SYN), Voisinage (VOIS) 106
5.5 Comparaison avec les résultats des autres systèmes 107
5.6 Abstraction dans les paquetages de Simulation d'événements discrets[109]. 108
5.7 Temps de réponse. 110
5.8 Nombre de Messages. 110
5.9 Rappel. 111
5.10 Précision. 111
5.11 Effet de la charge 113

Table des matières

1 Introduction **1**
- 1.1 Motivations . . . 2
- 1.2 Contributions . . . 5
- 1.3 Plan de la thèse . . . 6

2 État de l'art **7**
- 2.1 Introduction . . . 7
- 2.2 Le problème du partage de données . . . 7
- 2.3 Partage de documents . . . 8
- 2.4 Bases de données distribuées . . . 8
  - 2.4.1 Généralités . . . 8
  - 2.4.2 Traitement des requêtes . . . 9
  - 2.4.3 Médiation sémantique . . . 11
  - 2.4.4 Quelques projets de recherches . . . 14
  - 2.4.5 Limites des systèmes de gestion de bases de données distribuées . . . 15
- 2.5 Les réseaux Pair-à-Pair . . . 16
  - 2.5.1 Généralités . . . 16
  - 2.5.2 Indexation dans les réseaux P2P . . . 17
- 2.6 Partage de données dans un PDMS . . . 21
  - 2.6.1 Définition d'un PDMS . . . 21
  - 2.6.2 Stockage et Accès aux données . . . 23
  - 2.6.3 Topologie et routage de requête . . . 24
  - 2.6.4 Médiation de données sémantique dans les PDMS . . . 25
  - 2.6.5 Traitement des requêtes . . . 27
  - 2.6.6 Revue des systèmes existants . . . 28
- 2.7 Conclusion . . . 36

3 Médiation de données sémantique **37**
- 3.1 Introduction . . . 37
- 3.2 Notre contexte . . . 37
- 3.3 Architecture de SenPeer . . . 38
  - 3.3.1 Topologie du réseau . . . 38
  - 3.3.2 Architecture d'un pair . . . 40
  - 3.3.3 Architecture d'un super-pair . . . 41
  - 3.3.4 Communication entre (super-)pairs . . . 42
- 3.4 Le problème de la médiation . . . 43
- 3.5 Structure du modèle interne . . . 44
  - 3.5.1 Etiquettes des nœuds . . . 46
  - 3.5.2 Etiquettes des relations . . . 46
  - 3.5.3 Enrichissement sémantique . . . 47

3.5.4 Navigation dans un $sGraph$ . . . 49
3.6 Réconciliation sémantique . . . 50
3.6.1 Similarité globale . . . 51
3.6.2 Similarité linguistique . . . 51
3.6.3 Similarité de voisinage sémantique . . . 55
3.6.4 Génération des matrices de correspondance . . . 56
3.6.5 Illustration du processus d'appariement . . . 58
3.7 Bases de la topologie sémantique . . . 61
3.7.1 Expertise . . . 61
3.7.2 Création d'une communauté . . . 62
3.7.3 Adhésion à une communauté . . . 63
3.7.4 Graphe d'accointances . . . 63
3.7.5 Exemple . . . 64
3.7.6 Détection et gestion des fautes . . . 65
3.7.7 Sémantique de l'intégration de données . . . 66
3.8 Conclusion . . . 67

4 Traitement des requêtes **69**
4.1 Introduction . . . 69
4.2 Processus d'interrogation . . . 69
4.2.1 Idées de base du processus d'interrogation . . . 69
4.2.2 Sémantique de l'interrogation . . . 71
4.2.3 Format d'échange de requêtes SQUEL . . . 71
4.2.4 Navigation dans un graphe de requête SQUEL . . . 71
4.3 Mécanisme de routage sémantique . . . 73
4.3.1 Sélection basée sur l'expertise . . . 73
4.3.2 Algorithme de routage sémantique . . . 75
4.3.3 Illustration . . . 76
4.4 Reformulation de requêtes . . . 77
4.4.1 Le problème de la reformulation de requête . . . 77
4.4.2 Algorithme de reformulation de requête . . . 77
4.4.3 Illustration . . . 79
4.4.4 Attributs manquants . . . 79
4.4.5 Inclusion de requêtes . . . 80
4.4.6 Validité de la reformulation de requêtes . . . 82
4.5 Génération des plans de requêtes . . . 82
4.5.1 Vue d'ensemble . . . 82
4.5.2 Réécriture algébrique des requêtes . . . 84
4.6 Optimisation de requêtes . . . 86
4.6.1 Module d'optimisation de requêtes d'un (super-)pair . . . 86
4.6.2 Equivalences algébriques . . . 88
4.6.3 Énumération des plans alternatifs . . . 89
4.6.4 Modèle de coût . . . 93
4.6.5 Décomposition et distribution des plans de requêtes . . . 98
4.7 Conclusion . . . 99

5 Validation **101**
5.1 Introduction . . . 101
5.2 Évaluation de la découverte de correspondances . . . 101
5.2.1 Paramètres d'évaluation . . . 101
5.2.2 Résultats . . . 106
5.3 Evaluation du routage sémantique . . . 107
5.3.1 Outils de simulation . . . 107
5.3.2 Paramètres d'évaluation . . . 108
5.3.3 Mesures. . . . 109
5.3.4 Résultats expérimentaux . . . 109
5.4 Optimisation de requêtes . . . 112
5.5 Conclusion . . . 113

6 Conclusion **115**

Bibliographie **119**

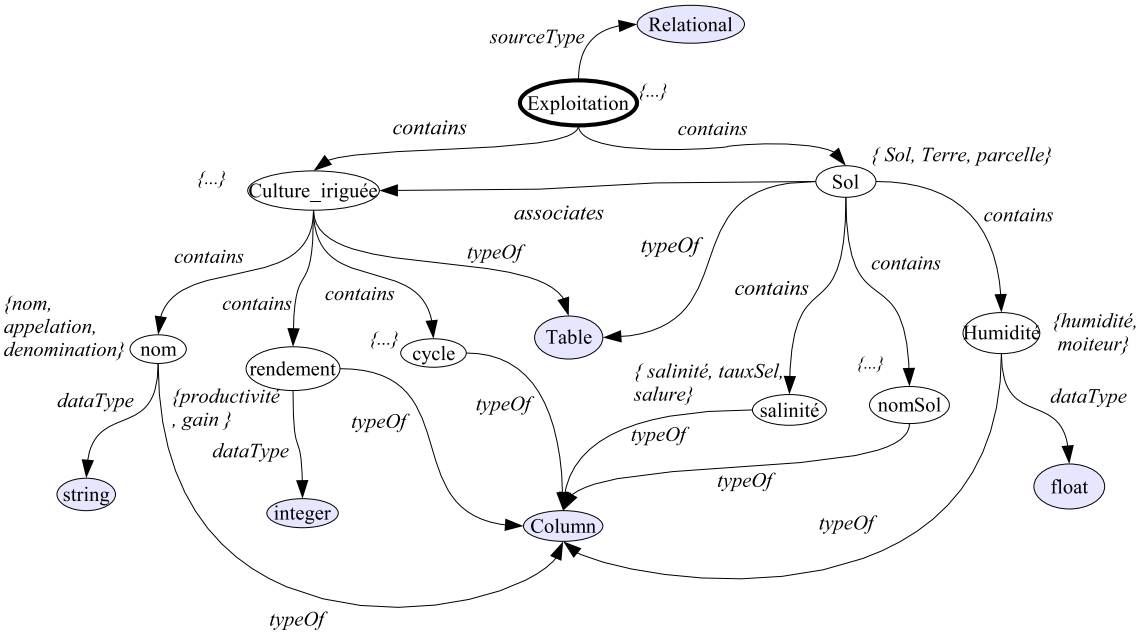
Liste des tableaux **129**

Table des figures **131**

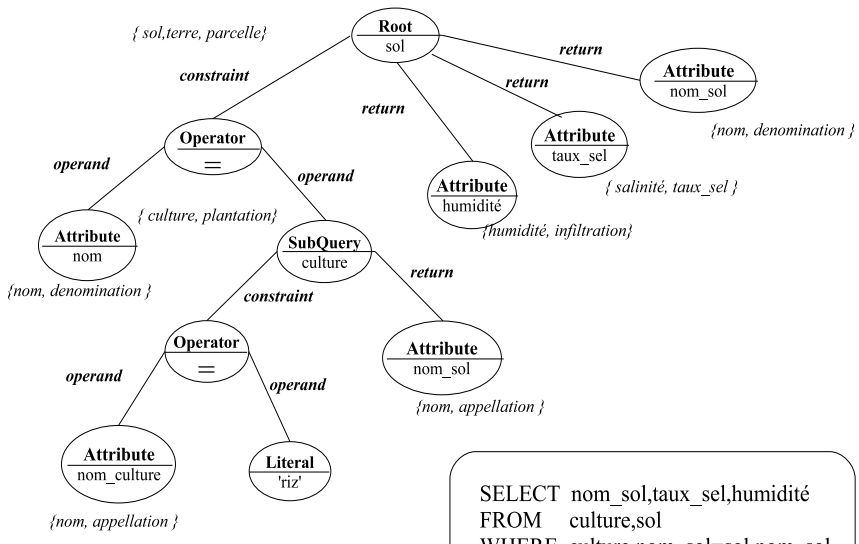
Table des matières **133**

Annexes

$sGraph$ partiel sur les cultures pratiquées sur les sols du bassin du fleuve.



Une requête SQL et la requête SQUEL correspondante.




```
SELECT  nom_sol,taux_sel,humidité
FROM    culture,sol
WHERE   culture.nom_sol=sol.nom_sol
AND     nom_cultur='riz'
```

sérialisation partielle du $sGraph$ précédent

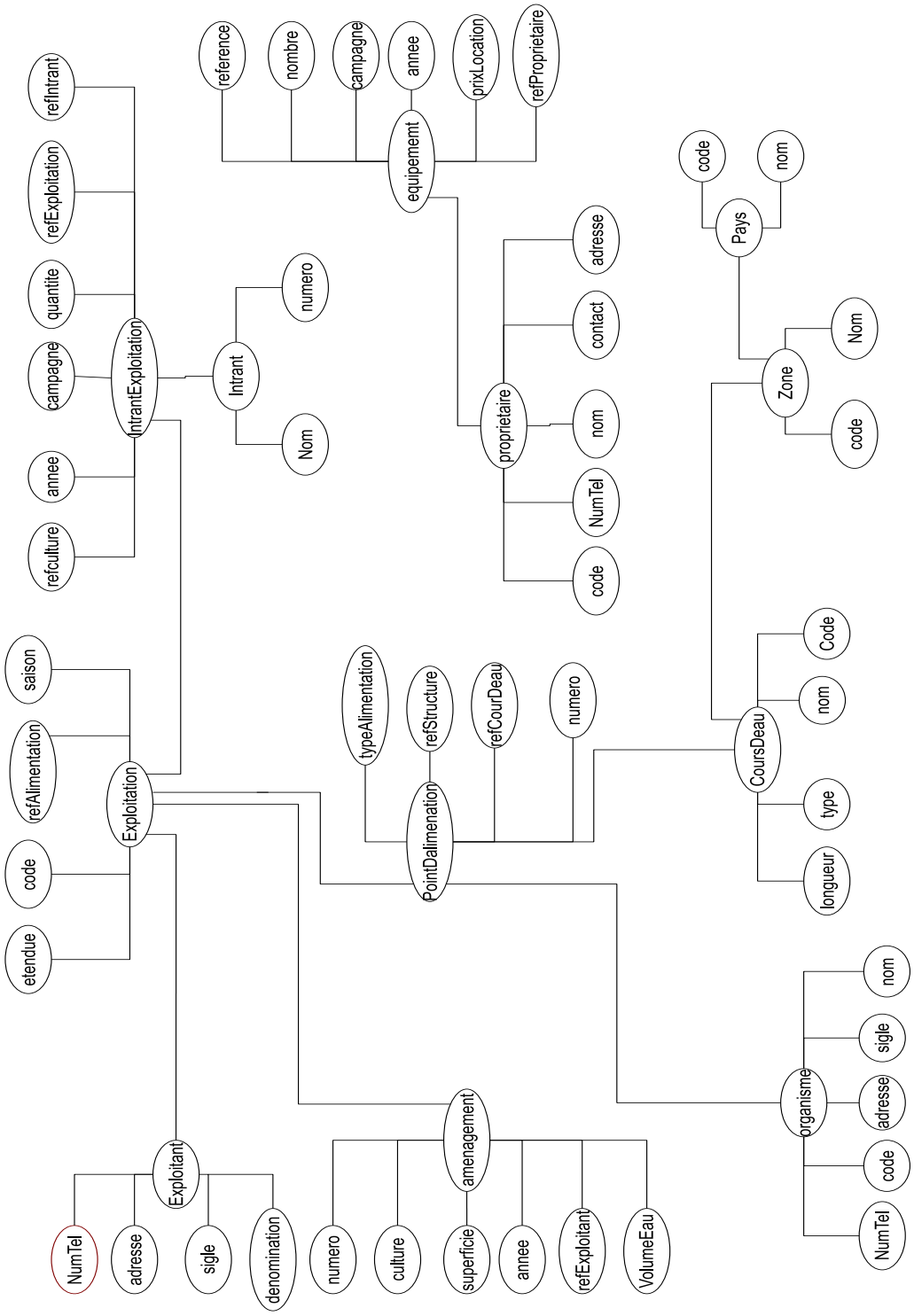
```
<?xml version="1.1"?>
<sGraph>
    <PeerName>SAED</PeerName>
        <sNode>
            <Name>Exploitation</Name>
            <SynSet>Exploitation,aménagement</SynSet>
            <SourceType>relational</SourceType>
            <Link name="contains"  destination="culture_irriguee"/>
            <Link name="contains"  destination="Sol"/>
        </sNode>
        <sNode>
            <Name>sol</Name>
            <SynSet>Sol,Terre,Parcelle</SynSet>
            <TypeOf>Table</TypeOf>
            <Link name="contains"  destination="humidite"/>
            <Link name="contains"  destination="nomSol"/>
            <Link name="contains"  destination="salinite"/>
        </sNode>
        <sNode>
            <Name>humidite<</Name>
            <SynSet>humidite,moiteur</SynSet>
            <TypeOf>Column</TypeOf>
            <DataType>integer</DataType>
            <Link name="contains"  source="sol"/>
        </sNode>
        <sNode>
            <Name>salinite</Name>
            <SynSet>salinite,tauxSel,salure</SynSet>
            <TypeOf>Column</TypeOf>
            <DataType>Integer</DataType>
            <Link name="contains"  source="sol"/>
        </sNode>
        <sNode>
            <Name>nomsol</Name>
            <SynSet>nom,appellation,denomination</SynSet>
            <TypeOf>Column</TypeOf>
            <DataType>String</DataType>
            <Link name="contains"  source="sol"/>
        </sNode>
        <sNode>
            <Name>culture_irriguee</Name>
            <SynSet>culture</SynSet>
            <TypeOf>Table</TypeOf>
            <Link name="contains"  destination="nom"/>
            <Link name="contains"  destination="rendement"/>
            <Link name="contains"  destination="cycle"/>
        </sNode>
        <sNode>
            <Name>rendement</Name>
            <SynSet>rendement,gain,productivité</SynSet>
            <TypeOf>Column</TypeOf>
            <DataType>Integer</DataType>
            <Link name="contains"  source="culture_irriguee"/>
        </sNode>
         <sNode>
            <Name>cycle</Name>
            <SynSet>cycle,périodicité</SynSet>
            <TypeOf>Column</TypeOf>
            <DataType>String</DataType>
            <Link name="contains"  source="culture_irriguee"/>
        </sNode>
        ..................................
</SGraph>
```

Sérialisation partielle de la requête SQUEL précédente

```
<?xml version="1.1"?>
<SQUELQuery>
       <QNode>
            <Name>sol</Name>
            <SynSet>sol,terre,parcelle</SynSet>
            <Type>ROOT</Type>
            <Link name="return"  destination="humidite"/>
            <Link name="return"  destination="nom_sol">
            <Link name="return"  destination="taux_sel"/>
            <Link name="constraint"  destination="="/>
        </QNode>
        <QNode>
            <Name>taux_sel</Name>
            <SynSet>taux_sel,salinité</SynSet>
            <Type>ATTRIBUTE</Type>
            <Link name="return"  source="sol">
        </QNode>
        <QNode>
            <Name>nom_sol</Name>
            <SynSet>nom,denomination</SynSet>
            <Type>ATTRIBUTE</Type>
            <Link name="return"  source="sol">
        </QNode>
        <QNode>
            <Name>humidite</Name>
            <SynSet>humidite,infiltration</SynSet>
            <Type>ATTRIBUTE</Type>
            <Link name="return"  source="sol">
        </QNode>
        <QNode>
            <Name>=</Name>
            <type>OPERATOR</Type>
            <Link name="operand"  destination="nom"/>
            <Link name="operand"  destination="culture"/>
        </QNode>
        <QNode>
            <Name>nom</Name>
            <type>ATTRIBUTE</Type>
            <Link name="operand"  source="="/>
        </QNode>
        <QNode>
            <Name>culture</Name>
            <Type>SUBQUERY</Type>
            <Link name="return"  destination="nom_sol"/>
            <Link name="constraint"  destination="="/>
            <Link name="operand"  source"="/>
        </QNode>
        <QNode>
            <Name>nom_sol</Name>
            <SynSet>nom,appellation</SynSet>
            <Type>ATTRIBUTE</Type>
            <Link name="return"  source="culture"/>
        </QNode>
        <QNode>
            <Name>=</Name>
            <Type>OPERATOR</Type>
            <Link name="operand"  destination="nom_culture"/>
            <Link name="operand"  destination="riz"/>
        </QNode>
        ..................

</SQUELQuery>}
```

Extrait du $sGraph$ de l'ADRAO



Médiation de données sémantique dans SenPeer, un système pair-à-pair de gestion de données

David Célestin FAYE

Résumé

La société de l'information demande un accès efficace à un ensemble d'informations qui sont souvent hétérogènes et distribuées. Dans le but d'un partage efficace de cette information, plusieurs solutions techniques ont été proposées. L'infrastructure Pair-à-Pair (P2P) est un paradigme émergent et offrant de nouvelles opportunités pour la mise en place de systèmes distribués à grande échelle. D'autre part, le concept de base de données distribuée a été introduit dans le but d'organiser une collection multiple de bases de données logiquement liées et distribuées sur un réseau d'ordinateurs. Récemment, les systèmes P2P de gestion de données communément appelés PDMS (*Peer Data Management System*) ont vu le jour. Ils combinent les avantages des systèmes P2P avec ceux des bases de données distribuées. Dans le but de contribuer à la recherche sur la gestion de données dans un contexte P2P, nous proposons le PDMS SenPeer. SenPeer suit une topologie super-pair basée sur l'organisation des pairs en communautés sémantiques en fonction de leur thème d'intérêt. Pour faciliter l'échange de données entre pairs nous établissons des processus de découverte de correspondances sémantiques et de reformulation de requêtes en présence de plusieurs modèles de données. Ces correspondances sémantiques, en combinaison avec les schémas des pairs sont à la base d'une topologie sémantique au dessus du réseau physique et utilisée pour un routage efficace des requêtes. Les requêtes sont échangées à travers un format commun d'échange de requête et un processus d'optimisation distribué permet de choisir le meilleur plan d'exécution de la requête en fonction des caractéristiques du PDMS. Une validation expérimentale par la mise en place d'un simulateur permet d'affirmer l'utilité et la performance des techniques proposées.

Mots-clés : Systèmes Pair-à-Pair, médiation de données sémantique, routage et optimisation de requêtes distribuées.

Abstract

The so-called information society needs an efficient access to the available information which is often heterogeneous and distributed. In order to make information sharing efficient, some technical solutions have been proposed. The concept of distributed database has been introduced in order to organize a collection of multiple and logically bound databases spread across a computer network. The Peer-to-Peer (P2P) infrastructure is an emergent paradigm offering new opportunities for the conception of large scale distributed systems. Recently the P2P data management systems (Peer Data Management System have appeared). They combine the advantages of the P2P systems with those of the distributed databases. In order to contribute to the research on data management in a P2P context, we propose the SenPeer PDMS. SenPeer is based on a super-peer topology organizing the peers into semantic communities according to their topics of interests. To facilitate data exchange in the presence of heterogeneous schemas and multiple data models, we establish semantic mapping discovery and query reformulation processes. The semantic mappings, in combination with the peer schemas are the basis of a semantic overlay, on top on the underlying physical network and which is used for intelligent query routing. Queries are exchanged through a query exchange format. A distributed optimisation process has the responsibility of finding the best execution plan for the query by taking into account the data distribution and the characteristics of the PDMS. We describe an experimental validation through a simulator to illustrate the feasibility and the performance of the proposed techniques

Keywords: Peer-to-Peer systems, semantic data mediation, distributed query routing and optimization.